SYBASE®
An SAP® Company

Security

Sybase Unwired Platform 2.2 SP02

DOCUMENT ID: DC01703-01-0222-01
LAST REVISED: January 2013

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

Contents

CHAPTER 1: Introduction to Security 1

Documentation Roadmap for Sybase Unwired Platform 1

Component Security 2

Communication Security 3

Device-to-Platform Communications 3

Unwired Server and Device Application Communications 4

Unwired Server and Device Push Notifications 5

Unwired Server and Data Tier Communications 5

Unwired Server and Sybase Control Center Communications 6

Unwired Server and EIS Communications 6

Unwired Server Nodes in Production Cluster Communications 7

Authentication and Access Security 7

Security Provider Plug-in Model 7

Security Configurations 8

CHAPTER 2: Security Quick Starts, Checklists, and Worksheets 11

Securing Data at Rest Quick Start 11

Securing Unwired Platform Data 12

Data at Rest Security Worksheet 12

Data at Rest Security Checklist 14

Securing Data in Motion Quick Start 16

Securing Synchronization 16

Securing Unwired Platform Runtime Component Communications 17

Enabling and Configuring Administration Encryption for Unwired Server 17
Data in Motion Worksheet 19
Securing Access Quick Start 21
Enabling Logins for Unwired Platform 21
Single Sign-on (SSO) Quick Start 22
Enabling Single Sign-on for DOE-C Packages 22
Enabling Single Sign-on for OData Applications 23
Enabling Single Sign-on for Mobile Business Object Packages 24
SSO Worksheet 25
SSO Checklists 25

CHAPTER 3: Server Security 27
Securing the Server Infrastructure 28
Handling Intrusion Detection/Prevention Software 28
Setting File System Permissions 30
Securing Platform Administration 30
Enabling Authentication and RBAC for Administrator Logins 31
Logging Into Sybase Control Center with an Installer-Defined Password 32
Making "Admin" Security Configuration Production-Ready 33
Disabling Authentication Caching and Increasing Log Levels 37
Validating the Production "Admin" Security Configuration 37
Enabling Authentication Caching and Reducing Log Levels 38
Resetting the supAdmin Password 38
Preparing SSL for HTTPS Listeners 40
Determining Certificate Requirements Based on Security Profile Chosen 40

Changing Installed Certificates Used for Unwired Server and Sybase Control Center HTTPS Listeners 41
Enabling and Configuring Administration Encryption for Unwired Server 45
Securing Multiple Domains 47
Determining a Tenancy Strategy 48
Benefits and Drawbacks of a Shared Security Configuration 49
Creating and Enabling a New Domain 49
Configuring Unwired Server to Securely Communicate With an HTTP Proxy 50
Enabling Authentication and RBAC for User Logins 51
Supported Providers and Credential Types 51
Considerations for Using E-mail Addresses as User Names 51
Authentication in Unwired Platform 52
Creating a Security Configuration for Device Users 53
Assigning Providers to a Security Configuration 53
Assigning Security Configurations to Domains, Packages, or Applications 64
Mapping Roles at the Global or Package Level ...64
SiteMinder Authentication with Sybase Unwired Platform 65
SiteMinder Client Authentication 65
Single Sign-on to a SiteMinder-protected EIS 66
Authentication Cache Timeout and Token Authentication 67
SiteMinder Web Agent Configuration for Sybase® Unwired Platform 68
Security Configuration to a SiteMinder-protected EIS 68
Single Sign-on Integration Across Client Applications ..71

Network Edge Single Sign-on Authentication 71
Single Sign-on Using NamedCredential 72
Propagate Single Sign-on Using ClientValuePropagatingLoginModule 72
Impersonation Prevention Using the checkImpersonation Property 74
Single Sign-on for SAP 74
Single Sign-on Authentication 75
Configuring X.509 Certificates for SAP Single Sign-on 76
SAP Single Sign-on and DOE-C Package Overview 78
SAP Single Sign-on and Mobile Business Object Package Overview 81
Creating Connections and Connection Templates 83
SAP Single Sign-on and Online Data Proxy Overview 90
Preparing Your SAP Environment for Single Sign-on 92
Security Configurations That Implement Single Sign-on Authentication 92
Creating Security Profiles to Enable Mutual Authentication for SAP 95
Enabling the HTTPS Port and Assigning the Unwired Server Security Profile 96
Distributing Single Sign-on Related Files in an Unwired Server Cluster 96
Stacking Providers and Combining Authentication Results 97
controlFlag Attribute Values 99
Stacking LoginModules in SSO Configurations . 100
Security Provider Issues 101
Encrypting Synchronization for Replication Payloads . . 101
End-to-End Encryption with TLS 102

Changing Installed Certificates Used for Encryption . .103
Modifying Default Synchronization Listener Properties with Production Values 104
Encryption Postrequisites 105
Encrypting Other Listeners for Unwired Server 105
Changing Keystore and Truststore Passwords 106
Defining Certificates for SSL Encryption 107
Creating an SSL Security Profile in Sybase Control Center 108
Enabling OCSP 110

CHAPTER 4: Data Tier Security 113

Securing the Data Infrastructure 113
Setting File System Permissions 113
Securing Backup Artifacts 114
Securing Data Tier Databases 114
Changing DBA Passwords for SQLAnywhere Databases in a Cluster Deployment 114
Changing DBA Passwords for SQLAnywhere Databases in a Single-Node Installation 116
Encrypting Data and Log Outputs 118

CHAPTER 5: DMZ Security 121

Relay Server as Firewall Protection 121
RSOE as the Unwired Server Protection 122
Relay Server and RSOE Communication Security 122
Configuring Connection Properties for Relay Server Components 123
Configuring Relay Server Connection Properties 123
Configuring Outbound Enabler Connection Properties 124

CHAPTER 6: Device Security 125

Limiting Application Access 125
Encrypting Device Data 126
Registering Applications, Devices, and Users 126
Registering Application Connections 127
Defining Applications 127
Locking and Unlocking a Subscription 127
Locking and Unlocking Application Connections 128
Securing Sensitive Data On-Device with DataVault 128
Enabling and Configuring a Password Policy for Data Vault Logins 130
Using Login Screens for Data Vaults 130
Provisioning Security Artifacts 131
Security Artifacts That Require Provisioning 131
Provisioning the Public RSA key from the Messaging Server for MBS Encryption 132
Establishing Encrypted Application Connections 132
Connecting to the TLS Relay Server Port with Client APIS 132
Connecting to the SSL Relay Server Port 133

CHAPTER 7: EIS Security 135
Securing EIS Operations: DCN and Push 135
Securing DCN Communications 136
Enabling Authorization of EIS Operations 137
SUP Roles to Support EIS Operations: SUP DCN User and SUP Push User 137
Setting Up Authorization with a Technical User Role Stored in a Repository 139
Setting Up Authorization with Certificate Validation 141
Setting Up Authorization with PreConfiguredUserLogin Values 144
Stacking Providers for DCN SSO Authentication 146

Related DCN Developer and Administrator Tasks 147
MBO Development for Data Change Notification 147
Hybrid App Development for Data Change Notification 147
Management and Monitoring of Data Change Notifications 148

CHAPTER 8: Security Monitoring and Issue Detection 149

Tools and Diagnostic Methodologies 149
Platform Security Monitoring 149
Reviewing System Monitoring Data 149
Common Analysis Scenarios 150
Access Denied Analysis 150
Checking the Security Log 150
Validating Security Setup 151

APPENDIX A: Security Reference 153

Security Provider Configuration Properties 153
LDAP Configuration Properties 153
NTProxy Configuration Properties 161
NoSecurity Configuration Properties 162
Certificate Authentication Properties 164
Certificate Validation Properties 166
HTTP Basic Authentication Properties 169
SAP SSO Token Authentication Properties 176
Preconfigured User Authentication Properties 177
Audit Provider Properties 178
DefaultAuditFilter Properties 179
FileAuditDestination Properties 182
XMLAuditFormatter Properties 183
Certificate and Key Management Utilities 183
Certificate Creation (createcert) Utility 184

Key Creation (createkey) Utility 186
Truststore and Keystore Properties 187
Port Number Reference 188

Index 193

CHAPTER 1 Introduction to Security

Mobility has changed the computing and network environments of today. Before mobility, enterprise security primarily focused on the firewall and limited access to digital assets to only those users authenticated within the enterprise information system (EIS).

A Sybase® Unwired Platform deployment introduces a multilayer approach to corporate security designed for mobility. This approach ensures that:

- Internal and external device users can securely connect to enterprise information systems.
- Every network link that transfers corporate information and every location that stores enterprise data guarantees confidentiality.

Before you can prepare for the scale and scope of activities required to secure Unwired Platform, learn which components you can secure, which communication streams you can protect, and how you can control access to mobile digital assets:

Documentation Roadmap for Sybase Unwired Platform

Sybase Unwired Platform documents are available for administrative and mobile development user roles. Some administrative documents are also used in the development and test environment; some documents are used by all users.

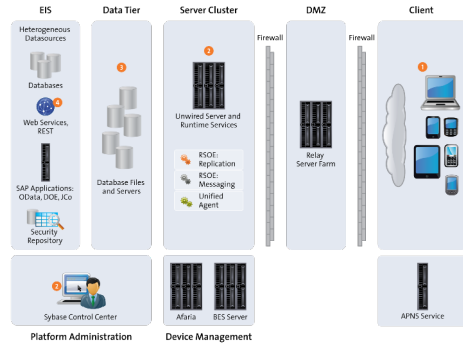
See *Documentation Roadmap* in *Fundamentals* for document descriptions by user role.

Check the Sybase® Product Documentation Web site regularly for updates: *http://sybooks.sybase.com/sybooks/sybooks.xhtml?id=1289&c=firsttab&a=0&p=categories* , then navigate to the most current version.

Component Security

Unwired Platform consists of multiple components that are installed on internal networks, primarily on the corporate LAN and the demilitarized zone (DMZ). Each component requires specific administration tasks to secure it.

Review this diagram to understand where platform components are installed, then review the table to understand how they are secured.



Numbers in this diagram identify various Sybase Unwired Platform security features. Some features are standards of the platform, while others are optional and up to the administrator or developer to implement.

Component	How Secured
1. Mobile application and local data	• Login screens • Encryption of local data • Initial provisioning • Remote administration and security features See *Device Security*.
2. Unwired Server and runtime data services	• Authentication of users and administrators • Enforcing device registration • Secure communication to subcomponents • Secure administration of server and services See *Server Security*.

Component	How Secured
3. Cache (CDB) and messaging database	• Encryption of data and logs • Supplying custom password during install; changing of installer-defined passwords supported See *Data Tier Security*.
4. Enterprise Information Servers (EIS)	• Secure connections • Secure data change notifications See *EIS Security*.

Communication Security

Secure Unwired Platform component communications to prevent packet sniffing or data tampering. Different combinations of components communicate with different protocols and different ports.

Note: As an alternative to reading all the topics on communication security, use *Port Number Reference* as a quick reference on all ports and information on how to change them.

See also

- *Port Number Reference* on page 188

Device-to-Platform Communications

Depending on your environment, devices typically connect to a Relay Server deployed to the DMZ (recommended). Alternatively, you can use a third reverse proxy or load balancers. However, in most Unwired Platform deployments, a Relay Server is the first line of defense to the platform by acting as a proxy for the device, and facilitating interactions with Unwired Platforms installed on the corporate LAN.

- For Relay Server connections, the traffic content depends on the payload protocol of the application:
  - Messaging communication encrypts the entire communication stream with a proprietary protocol and uses both HTTP and HTTPS protocol.
  - Replication communication uses both HTTP or HTTPS protocol.

  For Relay Server, configure the Web server host (IIS or Apache) to use a secure port, and use Sybase Control Center to configure Relay Server to use secure protocols, ports, and certificates in Sybase Control Center.

  On the client (in the case of mutual authentication), install certificates, and configure profiles to connect to Relay Server.

- Reverse proxies or load balancers require you to open firewall holes from the DMZ to Unwired Platform, but the same protocols described for messaging and replication still apply.

See also

- *Port Number Reference* on page 188

Unwired Server and Device Application Communications

Unwired Server communicates differently with replication, messaging, or Gateway applications.

- Replication applications – can use HTTP or HTTPS. By default, the data content in HTTP is unencrypted but compressed. HTTPS keeps the data confidential. For additional security, application developers can add end-to-end encryption (E2EE), in which all the data is encrypted between the device and Unwired Server. By default, the installer generates a key pair that is used by all installations of Unwired Platform. Therefore, you must replace these defaults to avoid compromising security. You can generate new ones, then replace these key pairs in Sybase Control Center. You must then provision the public client key to the device, and configure the device connection profile with the key location. Only then is data encrypted using an AES in cipher-block chaining mode; RSA handles the key exchange. See *Encrypting Synchronization for Replication Payloads*.
- Messaging applications, Hybrid Apps, and Online Data Proxy or OData applications – network traffic uses HTTP or HTTPS. Each HTTP message contains an encrypted message and follows this process:
  1. When the messaging server is installed, it generates an RSA key pair.
  2. When a device first contacts the server, the device retrieves the public key and the server uses it to secure all future communication. For performance reasons, only a small section of the data from device to server is encrypted with the public key. Other items of note:
     - Administrators can enable autoregistration by setting up an application connection template in Sybase Control Center. Automatic registration means that administrators need neither set up white lists nor generate single-use passwords. For details, see *Registering Application Connections* in *Sybase Control Center for Sybase Unwired Platform*.
     - For ODATA applications, developers can use Afaria® to preprovision the RSA public key to the client application.

       However, in non-Afaria environments or for non-ODATA applications excluding those for BlackBerry, Sybase requires that you initially install the messaging application, and connect directly to the messaging service on the corporate LAN (via Wi-Fi, cradle, and so on). Once initial registration is complete, the device can be used outside of the LAN by substituting the connection profile properties to use the Internet-accessible (typically, Relay Server) addresses. See *Provisioning Security Artifacts*

For BlackBerry devices, because the BES is already inside the LAN, the initial provisioning of the RSA public key is considered safe.

3. Registration adds the user name and authorization code to a white list. When the messaging client connects to the messaging server, it passes the user name, the activation code, and the device and application IDs to the server. The device ID is derived from the hardware. The user name, application ID, and device ID uniquely identify the registration.
4. The device identified by the DeviceID is permanently assigned to that user and added to the white list. For every future interaction, a communication session is initialized using the public key. For the remainder of the session, a rotating sequence of AES keys is used to yield better performance.

All data transferred between the device and the Messaging Server is encrypted in this manner.

See also

- *Provisioning Security Artifacts* on page 131
- *Encrypting Synchronization for Replication Payloads* on page 101
- *Port Number Reference* on page 188

Unwired Server and Device Push Notifications

Sometimes Unwired Server must asynchronously notify a device application of changes it should be aware of. The mechanism for transmitting a push notification depends on the device type.

The notification protocol depends on platform:

- For iOS devices, Unwired Server uses the APNS service. See *Apple Push Notification Properties* in *Sybase Control Center for Sybase Unwired Platform*.
- For Android devices, Unwired Server uses the GCM service. See *Android Push Notification Properties* in *Sybase Control Center for Sybase Unwired Platform*
- For BlackBerry devices, use the HTTP gateway push features of the MDS servers to deliver the notification. See *BlackBerry Push Notification Properties* in *Sybase Control Center for Sybase Unwired Platform*.
- For other types, use target change notifications (push notifications) in Unwired Server. See *Setting up Push Synchronization* in *System Administration*.

Unwired Server and Data Tier Communications

Unwired Platform uses Adaptive Server Anywhere (ASA) cache database for its data tier. It also connects to a cluster, monitor, and domain log database.

All communication streams between Unwired Platform and databases comprising are unencrypted, because they are exchanged on the corporate LAN. Nontheless, ensure that the local subnet is protected from network sniffing and file access.

See also

- *Port Number Reference* on page 188

Unwired Server and Sybase Control Center Communications

There are two different communication streams used to communicate with the Sybase Control Center administration tool: one for communications with the Sybase Control Center Web console, and one for the communications with the Sybase Control Center X.X Windows service.

Communications between Unwired Server and the Sybase Control Center X.X Windows service use IIOPS on port 2001 by default. While Unwired Platform installs a sample certificate to enable the use of IIOPS automatically, you should exchange the certificate with a production-ready one immediately following installation.

There are two self-signed certificates that need to be changed: one for Unwired Server, and one for Sybase Control Center.

See also

- *Changing Installed Certificates Used for Unwired Server and Sybase Control Center HTTPS Listeners* on page 41
- *Port Number Reference* on page 188

Unwired Server and EIS Communications

Secure communication between Unwired Server and any supported back-end depends on the direction of the interaction between these components.

- EIS to Unwired Server – can communicate only via the DCN feature. DCN uses HTTP or HTTPS and all requests are also authenticated via the DCN User role. Sybase discourages the use of HTTP for DCN because credentials could be intercepted by network sniffers in HTTP. To secure the channel:
  - The EIS developer uses HTTPS to construct and send DCN requests to the listener.
  - No action is required if the default configuration is used. If you want to create a custom DCN security profile in SCC, you must manage the certificates, then uses Sybase Control Center to configure a new HTTPS listener.

    Note: If you are connecting with Online Data Proxy or DOE-C, then each type of connection requires it's own security profile, and the DCN listener profile should not be used in this case.
- Unwired Server to EIS – Unwired Server can perform operation replays. The manner in which those replays are communicated depends on the EIS and whether or not the administrator secures this channel in Sybase Control Center:
  - REST/SOAP uses BASIC authorized over HTTP or HTTPS.
  - REST/SOAP for SAP® uses BASIC/SSO2/X.509 authentication over HTTP or HTTPS.

- JCo for SAP uses one of username/password, SSO2 tokens, or X.509 over SNC.
- JDBC uses driver specific mechanisms to encrypt traffic. Review your JDBC driver documentation to learn how to configure this.

For each of these EIS communication channels, you must configure the secure protocol. Otherwise, the user's credentials can potentially be exposed network sniffers. Once you have configured the secure channel, always ensure that EIS server certificates are imported into the Unwired Server truststore to allow this communication.

See also

- *Securing EIS Operations: DCN and Push* on page 135
- *Port Number Reference* on page 188

Unwired Server Nodes in Production Cluster Communications

Unwired Servers communicate with other Unwired Servers in the same cluster differently, depending on the type of communication performed.

- For replication synchronization, the servers use the secure replication ports to negotiate which server acts as the primary synchronization server.
- For the exchange JMS messages, servers use IIOPS.
- For other shared data or information, communicate indirectly using a shared databases on the data tier.

See also

- *Port Number Reference* on page 188

Authentication and Access Security

Authentication and role-based access control (RBAC) are core security features supported by all application types to control access to enterprise digital assets. Review key concepts of authentication and role-based access control in Unwired Platform.

Security Provider Plug-in Model

Implement authentication and access control with the Common Security Infrastructure (CSI) component. Use CSI to authenticate and authorize administrator, developer, and end-user operations. CSI has a service provider plug-in model that integrates with the customer’s existing security infrastructure.

Sybase Unwired Platform does not provide its own security systems for storing and maintaining users and access control rules, but delegates these functions to the enterprise’s existing security solutions. Security provider plug-ins for many common security solutions are included with Sybase Unwired Platform.

One of the service provider types, the login module, authenticates the user. The login module interface conforms to the Java Authentication and Authorization Service (JAAS). All of the

login modules in the Sybase Unwired Platform authenticate with user ID and password credentials. Multiple login modules — each of which links to a different security store — can be stacked. When the user logs in, each login module attempts authentication in the order specified in the CSI configuration definition. The authentication attempt stops iterating the sequence when authentication has been achieved or rejected.

For more information on using a custom security provider, see *Security API* in *Developer Guide: Unwired Server Runtime*.

Security Configurations

Sybase Unwired Platform does not provide proprietary security systems for storing and maintaining users and access control rules, but delegates these functions to the enterprise's existing security solutions.

A security configuration determines the scope of user identity, performs authentication and authorization checks, and can be assigned multiple levels (domain or package). Applications inherit a security configuration when the administrator assigns the application to a domain via a connection template.

Users can be authenticated differently, depending on which security configuration is used. For example, a user identified as "John" may be authenticated different ways, depending on the named security configuration protecting the resource he is accessing: it could be an MBO package, a DCN request, use of Sybase Control Center.

The anonymous security configuration provides unauthenticated user access, and is targeted to applications that do not require tight security.

The Agentry security configuration provides pass-through authentication to the Agentry Server for Agentry applications. The Agentry security configuration employs the NoSecLoginModule to allow user credentials to be sent to the Agentry server for authentication. Sybase Unwired Platform does not authenticate this security configuration.

Security configurations aggregate various security mechanisms for protecting Unwired Platform resources under a specific name, which administrators can then assign. Each security configuration consists of:

- A set of configured security providers. Security provider plug-ins for many common security solutions are included with the Sybase Unwired Platform.
- Role mappings (which are set at the domain and package level) that map logical roles to back end physical roles.

A user entry must be stored in the security repository used by the configured security provider to access any resources (that is, either a Sybase Control Center administration feature or an application package that accesses data sets from a back-end data source). When a user attempts to access a particular resource, Unwired Server tries to authenticate and authorize the user, by checking the security repository for:

- Security access policies on the requested resource

- Role memberships

CHAPTER 2 Security Quick Starts, Checklists, and Worksheets

Quick starts are task flows that identify important security setup activities in an Unwired Platform environment. Activities performed by the Unwired Platform administrator, may also require the collaboration or participation of mobile application developers, or Afaria, security, or database administrators, depending on the role distribution of your organization.

To assist you with your quick start activities, use worksheets and checklists as needed.

- Use worksheets to collect and document key decisions relating an activity.
- Use checklists to ensure you have prepared for an activity before starting it.

- *Securing Data at Rest Quick Start*

  Protecting data at the perimeter of a mobile enterprise is insufficieint and ignore a crucial vulnerability — sensitive data stored either on the device or on the runtime data tier are at risk from attackers who only need to find one way inside the network to access this confidential information.
- *Securing Data in Motion Quick Start*

  In a mobile environment, data in motion refers to the transfer of data between the source repository (EIS or backend), and the copies of data from that source as it traverses the perimeter of your organization into mobile networks or the Internet.
- *Securing Access Quick Start*

  Both Unwired Server and Sybase Control Center use Sybase Common Security Infrastructure (CSI). You configure how logins for administrators or devices users are processed.
- *Single Sign-on (SSO) Quick Start*

  Get started with SSO. Perform the activities required by the back-end EIS, using the checklists and worksheets provided for these workflows.

Securing Data at Rest Quick Start

Protecting data at the perimeter of a mobile enterprise is insufficieint and ignore a crucial vulnerability — sensitive data stored either on the device or on the runtime data tier are at risk from attackers who only need to find one way inside the network to access this confidential information.

Because perimeter defenses like firewalls and Relay Servers cannot protect stored sensitive data from this threat, you must use alternative means to prevent this type of exploitation.

Securing Unwired Platform Data

Secure data managed by the the Unwired Platform data tier. This includes all databases, including those that act as the Unwired Platform synchronization cache.

1. *Securing the Data Infrastructure*

   Secure data by first protecting the infrastructure on which it resides, then securing runtime databases.
2. *Securing Data Tier Databases*

   Secure all databases installed as the Unwired Platform data tier. You can change DBA passwords, grant DBA permissions to other users, and encrypt data and logs.
3. *Encrypting Device Data*

   Encrypting all data on the device client requires multiple techniques.
4. *Securing Sensitive Data On-Device with DataVault*

   (Not applicable to Hybrid Web Container) Developers should use a data vault with device applications to securely store “secrets” on the device. Data vaults are added using the DataVault API.

Data at Rest Security Worksheet

Record information about the data tier and its components. Refer to recorded information to streamline security tasks.

Non-System Administration Password (Installation Defined)

Administration Access	
SQLAnywhere DBA password	

Data Tier Servers

Data Tier Configuration Options	
Separate database and transaction log locations	
Data tier in failover cluster	

Data Tier Server Port Configuration		
Component	**Port**	**Password**
Cache database server		
Cluster database server		
LogData database server		

Data Tier Server Port Configuration		
Component	**Port**	**Password**
Messaging database server		n/a

Data Tier Failover Clusters

Data Tier Failover Cluster Configuration	
Shared cluster storage path (database files)	
Shared cluster storage path (transaction logs)	
Database server name	

Data Tier Data Paths

Data Tier Database File Locations	
Cache database data path	
Cluster database data path	
LogData database monitor data path	
LogData database domainlog data path	

Data Tier Transaction Logs

Data Tier Transaction Log File Locations	
Cache database log path	
Cluster database log path	
LogData database monitor log path	
LogData database domainlog log path	

Data Tier Encryption

Data Tier Algorithms Used	
Cache database data and log	
Cluster database data and log	
LogData database monitor data and log	

Data Tier Algorithms Used	
LogData database domainlog data and log	

Data Tier Backups

Data Tier Backup Policy	
Cache database backup location and frequency	
Cluster database backup location and frequency	
LogData database monitor backup location and frequency	
LogData database domain backup location and frequency	

File System Permissions

Component and Permission Levels	
Data tier permissions	

Data at Rest Security Checklist

Ensure you have secured platform and mobile data that is at rest, either on the corporate LAN or on client devices. Check activities off as you complete them.

Activity	Completed?
Set file system permissions on data tier hosts.	
Secured backup artifacts on data tier hosts.	
Encrypted data and log output for the data tier.	
Encrypted data on the device.	
Ensured that development has enabled a Data Vault for sensitive data.	

Note: It is incumbent on the application developer to retrieve and apply the Data Vault password policy that it gets from the server during application registration.

For example, in a Windows client using C#:

```
DataVault vault = null;
// handle first-run initialization - create vault, set password
policy
if (!DataVault.VaultExists("myVault"))
```

```
{
vault = DataVault.CreateVault("myVault", null, null);
vault.Unlock(null, null);
ApplicationSettings aps = app.ApplicationSettings;

if (aps.IsApplicationSettingsAvailable())
{
bool policyEnabled = (bool)
aps.GetBooleanProperty(ConnectionPropertyType.PwdPolicy_Enabled);
if (policyEnabled)
{
try
{
DataVault.PasswordPolicy oPasswordPolicy = new
DataVault.PasswordPolicy();
oPasswordPolicy.defaultPasswordAllowed = (bool)
aps.GetBooleanProperty(ConnectionPropertyType.PwdPolicy_Default_Pas
sword_Allowed);
oPasswordPolicy.minimumLength = (int)
aps.GetIntegerProperty(ConnectionPropertyType.PwdPolicy_Length);
oPasswordPolicy.hasDigits = (bool)
aps.GetBooleanProperty(ConnectionPropertyType.PwdPolicy_Has_Digits)
;
oPasswordPolicy.hasUpper = (bool)
aps.GetBooleanProperty(ConnectionPropertyType.PwdPolicy_Has_Upper);
oPasswordPolicy.hasLower = (bool)
aps.GetBooleanProperty(ConnectionPropertyType.PwdPolicy_Has_Lower);
oPasswordPolicy.hasSpecial = (bool)
aps.GetBooleanProperty(ConnectionPropertyType.PwdPolicy_Has_Special
);
oPasswordPolicy.expirationDays = (int)
aps.GetIntegerProperty(ConnectionPropertyType.PwdPolicy_Expires_In_
N_Days);
oPasswordPolicy.minUniqueChars = (int)
aps.GetIntegerProperty(ConnectionPropertyType.PwdPolicy_Min_Unique_
Chars);
oPasswordPolicy.lockTimeout = (int)
aps.GetIntegerProperty(ConnectionPropertyType.PwdPolicy_Lock_Timeou
t);
oPasswordPolicy.retryLimit = (int)
aps.GetIntegerProperty(ConnectionPropertyType.PwdPolicy_Retry_Limit
);
// SetPasswordPolicy() will always lock the vault to ensure the old
password
// conforms to the new password policy settings.
vault.SetPasswordPolicy(oPasswordPolicy);
vault.ChangePassword(null, null, pwd, null);
}
catch (DataVaultException dve)
{
Console.WriteLine("password not good enough? " + dve);
}
}
}
```

Securing Data in Motion Quick Start

In a mobile environment, data in motion refers to the transfer of data between the source repository (EIS or backend), and the copies of data from that source as it traverses the perimeter of your organization into mobile networks or the Internet.

To ensure data is protected along all communication channels, Sybase recommends that you use your existing PKI infrastructure to protect all Unwired Platform communications within and outside your corporate perimeter.

Securing Synchronization

Messaging data is automatically strongly encrypted over HTTP and HTTPS, using a private messaging payload protocol that is completely secure, once established. The message body is a JSON document. Therefore, only replication payloads require administrative intervention.

If your application uses the replication payload protocol, perform steps to secure the replication payload.

Note: When using OData, messaging data must be encrypted using HTTPS at a minimum to the Network Edge. Administrators must then protect data until it reaches Sybase Unwired Platform. For more information, see *Unwired Server and Device Application Communications* and *Developer Guide: OData SDK > OData SDK Components - General Description.*

1. *Changing Installed Certificates Used for Encryption*

   Unwired Server includes default certificates for all listeners. Since all installations use the same certificates by default, you must change these certificates with production-ready ones after you install Unwired Platform.
2. *Modifying Default Synchronization Listener Properties with Production Values*

   Once you have determined the degree of secure communication you require, you may need to modify default synchronization listener property values to disable one or more ports or synchronization protocols.
3. *Provisioning Security Artifacts*

   Typically, you must provision Unwired Platform security artifacts before applications can be used. The manner by which an artifact is provisioned depends on the artifacts themselves, the device types used, and your deployment environment.
4. *Establishing Encrypted Application Connections*

   Synchronization and messaging connection are encrypted by default. However, for replication connections that use E2EE, the client must be configured correctly to establish connections to the correct HTTP or HTTPS port.

Securing Unwired Platform Runtime Component Communications

There are two different ports that require encryption: the management port for Sybase Control Center (HTTPS), and the management ports used by Unwired Server (IIOPS).

You should also secure the server infrastructure to ensure that runtime binaries for the components performing the communication are protected from internal and external threats.

1. *Securing the Server Infrastructure*

   Before you can secure the runtime, you must first secure the underlying infrastructure. This activity prevents files from being tampered with on the host, also allows the Unwired Server to run with existing security mechanisms.
2. *Changing Installed Certificates Used for Unwired Server and Sybase Control Center HTTPS Listeners*

   Both Unwired Server and Sybase Control Center include default certificates that are used for these components' HTTPS listeners. Since all installations use the same certificates by default, you must change these certificates with production-ready ones after you install Unwired Platform. Unwired Server and Sybase Control Center share the same keystore and truststore (that is, SUP_HOME\Servers\UnwiredServer\Repository\Security\).
3. *Enabling and Configuring Administration Encryption for Unwired Server*

   Enable encryption to securely transfer data between the Unwired Server administration listener and Sybase Control Center.

Enabling and Configuring Administration Encryption for Unwired Server

Enable encryption to securely transfer data between the Unwired Server administration listener and Sybase Control Center.

You can create or change a security profile that saves SSL setup data for a particular server instance. Using the security profile, you associate a specific key with the encrypted port.

1. In the left navigation pane, expand the **Servers** folder and select a server.
2. Select **Server Configuration**.
3. In the right administration pane, click **General**.
4. Optional. If you want to create a new security profile, select **SSL Configuration**.
5. In the **Configure security profile table**:
   a) Enter a name for the security profile.
   b) Enter a certificate alias. This is the alias of a key entry in the keystore. Make sure the key password of this key entry is the same as the keystore password.
   c) Select an authentication level:

      If the security profile authenticates only the server, then only the server must provide a certificate to be accepted or rejected by the client. If the security profile authenticates both the client and the server, then the client is also required to authenticate using a

certificate; both the client and server will provide a digital certificate to be accepted or rejected by the other.

Profile	Authenticates	Cipher suite(s)
intl	server	• SA_EXPORT_WITH_RC4_40_MD5 • RSA_EXPORT_WITH_DES40_CBC_SHA
intl_mutual	client/server	• RSA_EXPORT_WITH_RC4_40_MD5 • RSA_EXPORT_WITH_DES40_CBC_SHA
simple	server	RSA_WITH_NULL_MD5 RSA_WITH_NULL_SHA
simple_mutual	client/server	RSA_WITH_NULL_MD5 RSA_WITH_NULL_SHA
strong	server	• RSA_WITH_3DES_EDE_CBC_SHA • RSA_WITH_RC4_128_MD5 • RSA_WITH_RC4_128_SHA
strong_mutual	client/server For example, this is the required option for mutual authentication of Unwired Platform and Gateway.	• RSA_WITH_3DES_EDE_CBC_SHA • RSA_WITH_RC4_128_MD5 • RSA_WITH_RC4_128_SHA
domestic	server	• RSA_WITH_3DES_EDE_CBC_SHA • RSA_WITH_RC4_128_MD5 • RSA_WITH_RC4_128_SHA • RSA_WITH_DES_CBC_SHA • RSA_EXPORT_WITH_RC4_40_MD5 • RSA_EXPORT_WITH_DES40_CBC_SHA • TLS_RSA_WITH_NULL_MD5 • TLS_RSA_WITH_NULL_SHA

Profile	Authenticates	Cipher suite(s)
domestic_mutual	client/server	• RSA_WITH_3DES_EDE_CBC_SHA • RSA_WITH_RC4_128_MD5 • RSA_WITH_RC4_128_SHA • RSA_WITH_DES_CBC_SHA • RSA_EXPORT_WITH_RC4_40_MD5 • RSA_EXPORT_WITH_DES40_CBC_SHA • RSA_WITH_NULL_MD5 • RSA_WITH_NULL_SHA

6. Use IIOPS in the Communication Ports sub-tab by selecting **Secure Management Port** (port 2001), and ensure that Sybase Control Center's Managed Resource properties match. By default, IIOPS is already configured between Unwired Server and Sybase Control Center.
7. Select the correct security profile name that provides the details for locating the correct certificates.
8. Save the changes and restart the server.

See also

• *Changing Installed Certificates Used for Unwired Server and Sybase Control Center HTTPS Listeners* on page 41

Data in Motion Worksheet

Record security setup options for data that moves from one point to another. Refer to recorded information to streamline security tasks.

Runtime Secure Communications: Ports and Certificates

Secure Port Configuration	
Server administration port	
Data change notification port	
Messaging port	
Replication port	
Unwired Server certificates and store location	

Secure Port Configuration	
Sybase Control Center certificates and store location	
Replication listener server certificate and end-to-end encryption key pairs and store location	

Non-System Administration Password (Installation Defined)

Administration Access	
Windows cluster administrator password	

File System Permissions

Component and Permission Levels	
Unwired Server host permissions	

Production Grade Security Providers for Administration

These providers enable role-based accesscontrol (RBAC) for administrators.

Providers for Administration Access	
Provider type	
Users or Groups for platform admin role mapping	
Users or Groups for domain admins role mapping	

Domains and Tenants

Domains and Tenancy Strategy	
Numbers of domains needed	
Names of domains	
Domain strategy employed	
Domain administrators assigned to each domain	
Security configurations assigned to each domain	

DCN Security

DCN SSL security	
Security profile name	
Authentication strength	
Certificate names and locations	
Security configuration	
SUP DCN User role mapping	

Securing Access Quick Start

Both Unwired Server and Sybase Control Center use Sybase Common Security Infrastructure (CSI). You configure how logins for administrators or devices users are processed.

CSI security providers perform these functions:

- **Authentication –** is performed using JAAS style LoginModules.
- **Authorization –** follows a role-based access control model.
- **Audit –** keeps an audit trail of authentication/authorization decisions made by CSI.
- **Role mapping –** when logical roles are used, allows you to map physical roles to logical ones.

Enabling Logins for Unwired Platform

Logins are configured differently for administrators and devices users. Sybase recommends that you keep the security configuration used by MBO packages separate from the “admin” configuration used by Unwired Platform administrators (platform administrators or domain administrators).

Therefore, determine your tenancy and domain strategy before configuring logins for administrators and users. With domains created, you can then easily assign security configurations or to the appropriate domain.

1. *Determining a Tenancy Strategy*

   Determine how many domains to create and how to distribute domain components. A strategic multitenant structure for the cluster balances system-availability considerations with administrative concerns.
2. *Enabling Authentication and RBAC for Administrator Logins*

   Role based access control (RBAC) for administrators is always performed by Unwired Server: Sybase Control Center automatically delegates administrator authentication to the providers configured for the "admin" security configuration on the "default" domain. When you install Unwired Platform, only the PreconfiguredUserLogin module is used for

the "admin" security configuration. To make the "admin" security configuration production-ready, you must initially log in using the administrator credentials defined with the installer, and replace the PreconfiguredUserLogin module with production-ready providers.

3. *Enabling Authentication and RBAC for User Logins*

   Enable authentication and role-based access control (RBAC) for device user logins by creating a new security configuration (that is, one that is distinct from the "admin" security configuration on the "default" domain), and mapping roles, then assigning it.

Single Sign-on (SSO) Quick Start

Get started with SSO. Perform the activities required by the back-end EIS, using the checklists and worksheets provided for these workflows.

Enabling Single Sign-on for DOE-C Packages

Enable single sign-on (SSO) over secure paths for Sybase SAP® Data Orchestration Engine Connector (DOE-C) packages.

1. *Preparing Your SAP Environment for Single Sign-on*

   Verify that the SAP® enterprise information system (EIS) is configured correctly to accept SSO connections from Unwired Server.

2. *Configuring X.509 Certificates for SAP Single Sign-on*

   Import, export, and generate the X.509 certificates that secure communication paths between Unwired Server and the SAP enterprise information system (EIS), and for client authentication, including single sign-on (SSO) with X.509 or SSO2 tokens.

3. *Deploying and Configuring DOE-C Packages*

   Unlike Hybrid App or MBO packages that use Sybase Control Center to deploy packages to Unwired Server, you must deploy the DOE-C package to specific domain using the DOE-C command line utility (CLU). Once deployed, the DOE-C package is visible and manageable from Sybase Control Center.

4. *Creating Security Profiles to Enable Mutual Authentication for SAP*

   Create security profiles and associate them with X.509 server certificates that can be used to establish secure connections between a client, Unwired Server, and the SAP EIS.

5. *Enabling the HTTPS Port and Assigning the Unwired Server Security Profile*

   Enable an HTTPS port for secure communication between Unwired Server and the SAP EIS.

6. *Enabling the DOE-C Connection*

   Configure the Sybase SAP® Data Orchestration Engine Connector (DOE-C) connection pool between Unwired Server and the SAP EIS. This is the port on which Unwired Server

communicates with the DOE, including forwarding subscriptions and allowing client operations to flow through to the DOE.

7. *Security Configurations That Implement Single Sign-on Authentication*

   Use the CertificateAuthenticationLoginModule authentication module to implement X.509 authentication or HttpAuthenticationLoginModule to implement SSO2.

8. *Provisioning the Public RSA key from the Messaging Server for MBS Encryption*

   If you are not using Afaria, you can install the client application, then connect to the corporate LAN using Wi-Fi or other method of your choosing in order to provision devices with required files. This allows you to seed public RSA keys to the device so that over-the-air connections to Unwired Server can be mutually-authenticated and you can minimize the possibility of a rogue server intercepting your initial synchronization and providing its own RSA public key.

See also

- *SAP Single Sign-on and DOE-C Package Overview* on page 78
- *Single Sign-on Authentication* on page 75

Enabling Single Sign-on for OData Applications

Enable single sign-on (SSO) over secure paths for OData applications.

1. *Preparing the SAP Gateway*

   Configure the SAP Gateway to push OData application data to Unwired Server, including configuring the RFC destination for HTTPS on the Gateway.

2. *Preparing Your SAP Environment for Single Sign-on*

   Verify that the SAP® enterprise information system (EIS) is configured correctly to accept SSO connections from Unwired Server.

3. *Using Keytool to Generate Self-Signed Certificates and Keys*

   Whenever possible, use a PKI system and a trusted CA to generate production-ready certificates and keys that encrypt communication among different Unwired Platform components. You can then use keytool to import and export certificate to the platform's keystores and truststores. Otherwise, you can also use keytool to generate self-signed certificates and keys.

4. *Configuring X.509 Certificates for SAP Single Sign-on*

   Import, export, and generate the X.509 certificates that secure communication paths between Unwired Server and the SAP enterprise information system (EIS), and for client authentication, including single sign-on (SSO) with X.509 or SSO2 tokens.

5. *Creating Security Profiles to Enable Mutual Authentication for SAP*

   Create security profiles and associate them with X.509 server certificates that can be used to establish secure connections between a client, Unwired Server, and the SAP EIS.

6. *Enabling the HTTPS Port and Assigning the Unwired Server Security Profile*

Enable an HTTPS port for secure communication between Unwired Server and the SAP EIS.

7. *Security Configurations That Implement Single Sign-on Authentication*

   Use the CertificateAuthenticationLoginModule authentication module to implement X.509 authentication or HttpAuthenticationLoginModule to implement SSO2.

8. *Provisioning Security Artifacts*

   Typically, you must provision Unwired Platform security artifacts before applications can be used. The manner by which an artifact is provisioned depends on the artifacts themselves, the device types used, and your deployment environment.

See also

- *SAP Single Sign-on and Online Data Proxy Overview* on page 90
- *Single Sign-on Authentication* on page 75

Enabling Single Sign-on for Mobile Business Object Packages

Enable single sign-on (SSO) over secure paths for mobile business object (MBO) packages.

1. *Single Sign-on for SAP MBO Package Prerequisites*

   Before implementing SSO for SAP MBO packages, configure the MBOs so client credentials can be propagated to the EIS and, if enabling SSO for a Hybrid App application, add the appropriate starting point.

2. *Preparing Your SAP Environment for Single Sign-on*

   Verify that the SAP® enterprise information system (EIS) is configured correctly to accept SSO connections from Unwired Server.

3. *Configuring X.509 Certificates for SAP Single Sign-on*

   Import, export, and generate the X.509 certificates that secure communication paths between Unwired Server and the SAP enterprise information system (EIS), and for client authentication, including single sign-on (SSO) with X.509 or SSO2 tokens.

4. *Creating Connections and Connection Templates*

   Create a new connection or connection template that defines the properties needed to connect to a new data source.

5. *Security Configurations That Implement Single Sign-on Authentication*

   Use the CertificateAuthenticationLoginModule authentication module to implement X.509 authentication or HttpAuthenticationLoginModule to implement SSO2.

6. *Provisioning Security Artifacts*

   Typically, you must provision Unwired Platform security artifacts before applications can be used. The manner by which an artifact is provisioned depends on the artifacts themselves, the device types used, and your deployment environment.

7. *Single Sign-on for SAP MBO Package Postrequisites*

After configuring SSO for SAP MBO packages on Unwired Server, install certificates on the mobile device and test them.

See also

- *SAP Single Sign-on and Mobile Business Object Package Overview* on page 81
- *Single Sign-on Authentication* on page 75

SSO Worksheet

Record SSO setup options and refer to recorded information to streamline SSO setup tasks.

SAP SSO Ports

Secure Port Configuration	
ICM HTTPS port	
SAP Gateway HTTPS port	

SAP Certificate Values

X.509 Certificate Properties	
User ID (DN) mapped to AS ABAP	
Certificate Alias	
Authentication strength	

SAP Connector Properties

SAP Connector Configuration	
Package deployment domain	
SNC certificate file	
SNC library location	

SSO Checklists

Mark activities you complete to ensure you have performed security tasks for SSO.

SAP MBO Checklist

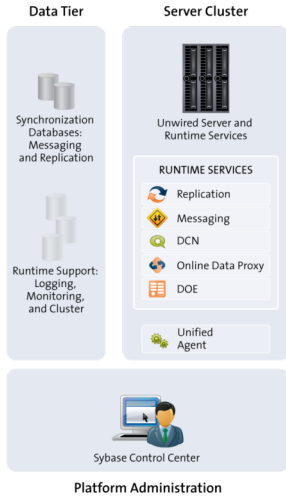
Activity	Completed?
Validate client IDs exist in SAP security repositories.	
(OData) Prepare the SAP Gateway.	

Activity	Completed?
(MBO) Validate that MBO package uses a JCo connector.	
(MBO) Validate that SAP function modules are exposed as Web services.	
(X.509) Use PKI to create certificates and keys for SSO.	
Enable SAP systems to communicate with Unwired Server (using either SSO2 tokens or X.509 certificates).	
Install required utilities and libraries onto Unwired Server hosts.	
Import SSL encryption certificates into Unwired Server stores.	
Create a security profile and enable the HTTPS port.	
(X.509) Import SSO certificates into Unwired Server stores.	
Create application templates to connect to the SAP data source (DOE, JCo/SNC for OData, or Web Services for MBO).	
Create security configurations and assign to the required packages/domains.	
Deploy packages to required domains.	
Provision devices with certificates.	
Validate and test connections to SAP backends.	

CHAPTER 3 Server Security

The Unwired Server provides data services to device clients by interacting with the data tier. The data tier is installed along with server tier components, to the internal corporate LAN.

Each runtime service uses its own communication port (secured and unsecured).



Secure the server runtime by performing activities that secure the infrastructure and administration of those components, in addition to enabling user authentication and secure communication.

1. *Securing the Server Infrastructure*

   Before you can secure the runtime, you must first secure the underlying infrastructure. This activity prevents files from being tampered with on the host, also allows the Unwired Server to run with existing security mechanisms.
2. *Securing Platform Administration*

   Use the Web-based console called Sybase Control Center to remotely and securely administer Unwired Platform.
3. *Enabling Authentication and RBAC for User Logins*

   Enable authentication and role-based access control (RBAC) for device user logins by creating a new security configuration (that is, one that is distinct from the "admin" security configuration on the "default" domain), and mapping roles, then assigning it.
4. *Encrypting Synchronization for Replication Payloads*

(Not applicable to Online Data Proxy) By default, the Unwired Server replication listener is configured to use TLS for end-to-end encryption (E2EE) on HTTP and HTTPS ports, and SSL for encryption on HTTPS ports.

5. *Encrypting Other Listeners for Unwired Server*

   By default all other Unwired Platform listeners are encrypted using SSL. However, if you need to modify this configuration, review these steps.

Securing the Server Infrastructure

Before you can secure the runtime, you must first secure the underlying infrastructure. This activity prevents files from being tampered with on the host, also allows the Unwired Server to run with existing security mechanisms.

1. *Handling Intrusion Detection/Prevention Software*

   A personal firewall, or intrusion detection/prevention software (IPS or IDPS), can cause Unwired Platform components to malfunction or not function at all. Unwired Platform uses regular IP communication between components on the primary network interface of a computer, even when all components are installed on the same host.

2. *Setting File System Permissions*

   Unwired Platform runs as a collection of Windows services. During installation, you are prompted with a Logon as request. The credentials collected are then used to run the service under that account. In a cluster installation, the same Windows user would be configured for all installations and respective Window services that are subsequently installed.

See also

- *Securing Platform Administration* on page 30

Handling Intrusion Detection/Prevention Software

A personal firewall, or intrusion detection/prevention software (IPS or IDPS), can cause Unwired Platform components to malfunction or not function at all. Unwired Platform uses regular IP communication between components on the primary network interface of a computer, even when all components are installed on the same host.

If the local network interface is secured by intrusion detection/prevention software (IPS or IDPS, for example, McAfee Host Intrusion Prevention software or equivalent), you must configure the security software to allow all network communication between Unwired Platform components.

For a single-node installation of all of the Sybase Unwired Platform components, try one of these options to work around the limitations imposed by the host intrusion prevention software and policy settings, without violating any security policy, until the settings of your security software are adjusted to the needs of Unwired Platform.

Choose an option:

- Removing the host machine from the network – this option ensures that all interconnections between Sybase Unwired Platform components are treated as local traffic and is not be flagged as incoming connections from external sources, thereby causing connection failures due to security policy setting. This option is suitable when you use your laptop in a network other than your corporate network, and want to demonstrate a mobile solution using a simulator or emulator with all components running on the same machine. To use this option:
  1. Stop the Sybase Unwired Platform services in the correct order. See *Methods for Starting and Stopping Unwired Platform* in *System Administration*.
  2. Disconnect the host from all networks.
  3. Restart Sybase Unwired Platform services in the correct order.
  4. Change the Sybase Control Center URL link to use "localhost" or <*yourhostname*> as the host name, instead of the original fully qualified host name of the machine that included the domain name (for example: `https://localhost:8283/scc`, or `https://`*`yourhostname`*`:8283/scc`). Accept any security warnings to connect to Sybase Control Center.
- Connecting the host to the corporate network – this option ensures that all interconnections among Sybase Unwired Platform components are internal to your corporate network and validated against the corporate network security policy. The option of connecting to corporate network through VPN is especially suitable when you use your laptop in a network other than your corporate network, and want to demonstrate a mobile solution using your physical devices, and need outgoing connections to a backend Enterprise Information System (EIS) or Relay Server (Sybase Hosted Relay Server or otherwise).
  1. Stop the Sybase Unwired Platform services in the correct order. See the *Methods for Starting and Stopping Unwired Platform* topic in *System Administration*.
  2. Reconnect the host to your corporate network directly or through corporate VPN, to ensure that the corporate network security policy applies.
  3. Restart Sybase Unwired Platform services in the correct order.
  4. Change the Sybase Control Center URL link to use "localhost" or <*yourhostname*> as the host name, instead of the original fully qualified host name of the machine that included the domain name (for example: `https://localhost:8283/scc`, or `https://`*`yourhostname`*`:8283/scc`). Accept any security warnings to connect to Sybase Control Center.
- So required internal component communication ports are not blocked, configuring the firewall software to allow connections to the ports the Unwired Platform uses. For information about what ports you must accommodate, see *Unwired Platform Port Accommodation* in the *Landscape Design and Integration* guide.

Always check for the latest available patches and updates for your Unwired Server version on *http://downloads.sybase.com/swd/base.do?client=support* (logon account required).

Setting File System Permissions

Unwired Platform runs as a collection of Windows services. During installation, you are prompted with a **Logon as** request. The credentials collected are then used to run the service under that account. In a cluster installation, the same Windows user would be configured for all installations and respective Window services that are subsequently installed.

You can restrict permissions after installation by removing most users and groups from the Unwired Platform installation directory.

1. Open File Explorer.
2. Right-click *SUP_HOME*, and click **Properties**.
3. On the **Security** tab, click **Advanced**.
4. Unselect **Inherit from parent the permission entries that apply to child objects. Include these with entries explicitly defined here.**.
5. In the confirmation pop-up, choose **Copy**, then select **Replace permission entries on all child objects with entries shown here that apply to child objects.**.
6. In the table of Permission entries, remove all users except the user account that was configured as the Logon user for the Windows services. If another user is responsible for some activities extend the necessary permissions to this administrator. For example, if the individual is only reading log files, you may choose to limit permissions to read only.
7. Click **OK**.

Securing Platform Administration

Use the Web-based console called Sybase Control Center to remotely and securely administer Unwired Platform.

Sybase Control Center relies on a Windows service called Sybase Control Center X.X that runs on each Unwired Server on the cluster. The service handles communication between Sybase Control Center and Unwired Server runtime components.

1. *Enabling Authentication and RBAC for Administrator Logins*

   Role based access control (RBAC) for administrators is always performed by Unwired Server: Sybase Control Center automatically delegates administrator authentication to the providers configured for the "admin" security configuration on the "default" domain. When you install Unwired Platform, only the PreconfiguredUserLogin module is used for the "admin" security configuration. To make the "admin" security configuration production-ready, you must initially log in using the administrator credentials defined with the installer, and replace the PreconfiguredUserLogin module with production-ready providers.
2. *Resetting the supAdmin Password*

You can manually reset the current platform administration password.

3. *Preparing SSL for HTTPS Listeners*

   SSL requires that two self-signed certificates installed by default need to be replaced: one for Unwired Server, and one for Sybase Control Center. Once the certificates have been imported, you can configure the security profile for these components in Sybase Control Center.

4. *Securing Multiple Domains*

   To prevent role mapping leaks between multiple tenant domains, configure domains and assign shared security configurations.

5. *Configuring Unwired Server to Securely Communicate With an HTTP Proxy*

   If you want Unwired Server connect to an HTTP Proxy, you can set connection properties for it when you optimize Unwired Server performance.

See also

- *Securing the Server Infrastructure* on page 28
- *Enabling Authentication and RBAC for User Logins* on page 51

Enabling Authentication and RBAC for Administrator Logins

Role based access control (RBAC) for administrators is always performed by Unwired Server: Sybase Control Center automatically delegates administrator authentication to the providers configured for the "admin" security configuration on the "default" domain. When you install Unwired Platform, only the PreconfiguredUserLogin module is used for the "admin" security configuration. To make the "admin" security configuration production-ready, you must initially log in using the administrator credentials defined with the installer, and replace the PreconfiguredUserLogin module with production-ready providers.

The PreconfiguredUserLoginModule does not enforce password strength or change policies that would typically be in place for a production environment. Therefore, substitute the PreconfiguredUserLogin module with one that is suitable for a production environment. Subsequent logins are then performed with user credentials assigned to the platform or domain administrator role.

The “admin” security configuration is used to authenticate and authorize administrative users. The “admin” security configuration is on the “default” domain. The “default” domain is where critical runtime configuration artifacts exist.

Sybase recommends that you restrict the use of the “admin” security configuration on the “default” domain to administration authentication only. The “admin” security configuration should not be used for other domains.

1. *Logging Into Sybase Control Center with an Installer-Defined Password*

   The person acting as platform administrator logs in to Sybase Control Center for the first time after installation.

2. *Making "Admin" Security Configuration Production-Ready*

   Replace the default PreConfiguredUserLoginModule with new production-ready providers.
3. *Disabling Authentication Caching and Increasing Log Levels*

   Temporarily disable administrator authentication caching, so the new provider can be validated. Also increase log levels to capture more detailed events in case you need to troubleshoot problems.
4. *Validating the Production "Admin" Security Configuration*

   Once LDAP has been added as a provider to the "admin" security configuration for Unwired Server, you can test the login before removing the PreconfiguredUser login module from both components' configurations.
5. *Enabling Authentication Caching and Reducing Log Levels*

   Re-enable administrator authentication caching as required by your environment, and reduce log levels to a value more appropriate for normal security operations.

See also

- *Resetting the supAdmin Password* on page 38
- *SUP Platform Administrator* on page 34
- *SUP Domain Administrator* on page 35
- *SUP Helpdesk* on page 35
- *Authentication in Unwired Platform* on page 52

Logging Into Sybase Control Center with an Installer-Defined Password

The person acting as platform administrator logs in to Sybase Control Center for the first time after installation.

During installation, the person installing Unwired Platform defines a password for the supAdmin user. This password is used to configure the Preconfigured login module that performs the administrator authentication.

Note: This installer-defined password is not intended to be a permanent administrator credential. You must replace this module with a production-grade authentication module, typically LDAP.

1. Launch Sybase Control Center.
2. Enter `supAdmin` for the user name and type the <*`supAdminPwd`*> for the password. Note that the user name is case sensitive.
3. Click **Login.**
4. Open the Unwired Platform perspective and authenticate with Unwired Server using the same credentials used to log into Sybase Control Center.

Making "Admin" Security Configuration Production-Ready

Replace the default PreConfiguredUserLoginModule with new production-ready providers.

Note: Please note these restrictions before beginning:

- For LDAPLoginModule, special characters (for example, `, = : ' " * ? &`) cannot be used in the user name defined with the Authentication Filter property. This same property also does not support Chinese or Japanese characters in the user name and password properties.
- For PreConfiguredUserLoginModule, the User Name property also cannot contain the same special characters (that is, `, = : ' " * ? &`).

---

1. *Adding a Production-Grade Provider*

   Modify the "admin" security configuration to add a production-grade provider, typically an LDAPLoginModule. Most companies use an LDAP directory to maintain internal user accounts. This module integrates with most LDAP servers including Active Directory.
2. *Mapping Unwired Platform Logical Roles to Physical Roles*

   All administrative users and their passwords are managed in the enterprise security repository. In order for administrative users to have access to the Unwired Server in a production environment, you must map the SUP default logical roles to the corresponding physical roles or groups in the security repository.

See also

- *Disabling Authentication Caching and Increasing Log Levels* on page 37

Adding a Production-Grade Provider

Modify the "admin" security configuration to add a production-grade provider, typically an LDAPLoginModule. Most companies use an LDAP directory to maintain internal user accounts. This module integrates with most LDAP servers including Active Directory.

Prerequisites

Determine what values are needed for the login module properties in Unwired Platform by gathering this information from the security provider you will be using. For example, for an LDAP login module you need values for the providerURL, serverType, bind user, bind password, search base and so on.

Task

Configure the "admin" security configuration on the "default" domain to authenticate only platform and domain administrators, and help desk operators. All Sybase Control Center users must belong to the "admin" security configuration. It is recommended that you create custom security configurations for Sybase Unwired Platform application users and assign these security configurations to domains and MBO packages.

1. In the navigation pane of Sybase Control Center, expand the **Security** folder, then click the security configuration named **admin**.
2. In the administration pane, click the **Authentication** tab.
3. Add an LDAPLoginModule, configuring the providerURL, serverType, bind user, bind password, search base, and other properties determined by you and the LDAP administrator.
4. Add a `ControlFlag` attribute for the configured LDAP login module, and set the value to `sufficient`.
5. Make the LDAP module the first module in the list.

   Note: Do not remove the PreConfiguredUser login module from the list of login modules used by the "admin" security configuration until the LDAP login module has been tested.
6. Select the **General** tab, select **Validate**, then **Apply**.
7. Click **OK**.

See also

- *Preconfigured User Authentication Properties* on page 177
- *LDAP Security Provider* on page 56
- *LDAP Configuration Properties* on page 153

Mapping Unwired Platform Logical Roles to Physical Roles

All administrative users and their passwords are managed in the enterprise security repository. In order for administrative users to have access to the Unwired Server in a production environment, you must map the SUP default logical roles to the corresponding physical roles or groups in the security repository.

Default SUP Administrator Roles

Unwired Platform roles are logical roles that are built into the system by default. To enable role-based access to the administrative interface, configure mapping of the Unwired Platform Administrator and Unwired Platform Domain Administrator logical roles to roles that exist in the security repository used for administrative authentication and authorization. In addition, you can configure the Unwired Platform Helpdesk role, which provides read-only access to the administrative interface.

For a list of the typical tasks performed by the logical administrator roles, see *Platform Administration Roles and Tasks* in *Sybase Control Center for Sybase Unwired Platform*.

SUP Platform Administrator

Platform administrators interact with Unwired Platform to perform high-level, cluster-wide management.

A platform administrator can perform all administrative operations in the Unwired Platform administration console, including domain administration for all domains.

Note: The terms "Unwired Platform (platform) administrator" is used in all documentation to refer to the user with "Sybase Unwired Platform Administrator" role.

See also

- *Enabling Authentication and RBAC for Administrator Logins* on page 31

SUP Domain Administrator

Domain administrators interact with Unwired Platform to manage packages, server connections, security configurations, and role mappings in specific domains. A domain is a logical partition used to isolate and manage runtime artifacts for a particular tenant.

The domain administrator can administer only the domain to which he or she is assigned. The domain administrator is granted access on a per-domain basis by the platform administrator.

The logical role for the domain administrator is "SUP Domain Administrator." The term "domain administrator" is used in all documentation to refer to the user with the "SUP Domain Administrator" role.

See also

- *Enabling Authentication and RBAC for Administrator Logins* on page 31

SUP Helpdesk

Help desk operators interact with Unwired Platform to review system information to determine the root cause of reported problems. Help desk operators only need to view information, not change it.

Help desk operators have read-only access to all administration information in the Unwired Platform administration console. They cannot perform any modification operations on administration console tabs, and cannot save changes made in dialogs or wizards.

The logical role for the help desk operator is "SUP Helpdesk." The term "help desk operator" is used in all documentation to refer to the user with the "SUP Helpdesk" role.

See also

- *Enabling Authentication and RBAC for Administrator Logins* on page 31

Gathering Provider Group Information

Production environments rely on a production-grade security provider (commonly an LDAP directory) to authenticate administrators. To map the SUP default logical roles to the corresponding physical roles in the security provider, you must understand how the provider organizes users into groups.

Consider which users need to be in the SUP Administrator, SUP Domain Administrator, and SUP Helpdesk roles, then identify or create groups in your provider that corresponding to these roles.

Note: If you have installed an earlier version of Unwired Platform as part of a development deployment, you may have an OpenDS LDAP server running in your environment, and both Unwired Platform and Sybase Control Center may be using this directory. Sybase no longer uses this directory and strongly encourages you to use a different LDAP directory.

1. Evaluate existing groups.

   If there are existing groups that seem to already contain the right subjects that correspond to SUP Administrator, SUP Domain Administrator, and SUP Helpdesk platform roles, you can use those groups. The names need not be exact, as you can map them in Sybase Control Center to address any differences.
2. If no sufficient group exists, add them for Unwired Platform.
3. Add subjects to these groups to assign Unwired Platform corresponding permissions.
4. Determine what values are needed for the login module properties in Unwired Platform.
   For example, for an LDAP login module you need values for the providerURL, serverType, bind user, bind password, search base and so on.

Mapping Default Administrator Roles

In order for administrators to be able to access the Unwired Server, you must map the default Sybase Unwired Platform logical roles to the corresponding physical roles in the security provider. You perform the mapping for the "default" domain in the "admin" security configuration.

Use Sybase Control Center to map these logical roles to the appropriate physical roles or groups in the underlying security provider.

1. Open Sybase Control Center.
2. In the left navigation pane, expand **Domains**.
3. Expand the **default** domain.
4. Expand the Security folder and click the **admin** security configuration.
5. For each logical role that you want to map:
   - If logical role exactly matches the role name in the security provider repository, select **AUTO**.
   - If the logical role differs from the role name in the security provider repository, select **Map Roles**.
6. To map the logical role to the physical role in the Role Mapping dialog:
   a) Select the physical role that you want to map the logical role to in **Available roles**.
   b) Click **Add**.

Once a logical role has been manually mapped, the mapping state changes to MAPPED.

Note: You can also map roles for the "default" security configuration through the **Security** node. For details see *Mapping Roles for a Security Configuration* in *Sybase Control Center for Sybase Unwired Platform* online help.

Once a logical role has been manually mapped, the mapping state changes to MAPPED.

Disabling Authentication Caching and Increasing Log Levels

Temporarily disable administrator authentication caching, so the new provider can be validated. Also increase log levels to capture more detailed events in case you need to troubleshoot problems.

1. Disable authentication caching:
   a) In the left navigation pane, expand the **Security** folder.
   b) Select the "admin" security configuration, and display its properties.
   c) In the right administration pane, select the **Settings** tab.
   d) Set the cache timeout value to 0, which tells Unwired Server to not cache results.
   e) Click **Save**.
2. Increase log levels to a more sensitive value:
   a) In the left navigation pane, select **Configuration**.
   b) In the right administration pane, click the **Log Settings** tab.
   c) For the security log component, set the log level to DEBUG, which provides detailed system information, warnings, and all errors.
   d) Click **Save**.

See also

- *Making "Admin" Security Configuration Production-Ready* on page 33
- *Authentication Cache Timeouts* on page 54

Validating the Production "Admin" Security Configuration

Once LDAP has been added as a provider to the "admin" security configuration for Unwired Server, you can test the login before removing the PreconfiguredUser login module from both components' configurations.

1. Log in to Sybase Control Center using the login values of an LDAP user that is in an LDAP group mapped to the "SUP Administrator" logical role.
2. If the login succeeds:
   a) Remove PreconfiguredUser login module from Unwired Server and Sybase Control Center setup locations.
   b) Reduce the logging levels for both Unwired Server and Sybase Control Center to Info or Warn.
   c) Restart Unwired Server.
3. If the login fails:

a) Check the Sybase Control Center log in *SCC_HOME*\log\agent.log to see if authentication failed with this component.
b) If no issues are identified, continue checking with the Unwired Server log in *SUP_HOME*\Servers\UnwiredServer\logs*<ClusterName>*-server.log.
c) If no issues are immediately apparent, review security issues documented in *Troubleshooting* for possible resolution guidelines.

Enabling Authentication Caching and Reducing Log Levels

Re-enable administrator authentication caching as required by your environment, and reduce log levels to a value more appropriate for normal security operations.

1. Enable authentication caching:
   a) In the left navigation pane, expand the **Security** folder.
   b) Select the "admin" security configuration, and display its properties.
   c) In the right administration pane, select the Settings tab.
   d) Set the cache timeout value in seconds. The default is 3600 seconds.

      The **Authentication cache timeout** determines how long authentication results should be cached before the administrator is required to reauthenticate.
   e) Click **Save**.
2. Return log levels to a less sensitive value:
   a) In the left navigation pane, select **Configuration**.
   b) In the right administration pane, click the **Log Setting** tab.
   c) For the security log component, set the log level to WARN.
   d) Click **Save**.

See also

- *Authentication Cache Timeouts* on page 54

Resetting the supAdmin Password

You can manually reset the current platform administration password.

This procedure is for Sybase Unwired Platform version 2.x.x or later.

Note: You must contact your IT department or administrator before changing or resetting the password. Only the person who has the right permission to change these files should perform this procedure.

1. Open the Unwired Server default.xml file, located in *SUP_HOME*\Servers\UnwiredServer\Repository\CSI\conf and modify this line:

```
<options encrypted="false" name="password"
value="{TXT:}s3pAdmin" />
```

In this example, you are setting the new password to `s3pAdmin`. You can replace this password with any password you choose. Do not remove the `{TXT:}` prefix to the password.

Note: In this example, password encryption is set to false. Disregard this value; you will configure encryption correctly in step 6.

2. Save the file.
3. Restart Unwired Server and Sybase Control Center.
4. Log in to Sybase Control Center using supAdmin as the new password.
5. When login succeeds, Sybase Control Center opens the management view on the local Unwired Server.
6. In Sybase Control Center, expand the **Security** node:
   a) Click **admin**.
   b) Click **Authentication** and select **PreConfiguredUserLoginModule** for the supAdmin user.
   c) Click **Properties**, and enter the new password. By resupplying the password here, the file is overwritten using the correct syntax.
   d) Click **Save**.
   e) When you see the warning message, click **OK**, then click the **General** tab.
   f) Click **Apply**.

      You see another warning.
   g) Click **OK**.

      Configuration files are rewritten using the values you entered. When the process completes, you see a `Successfully saved` message.
7. Login again to Sybase Control Center using the supAdmin login and the new password (in this example, s3pAdmin).
8. Go to the `...\UnwiredServer\Repository\CSI` folder and verify `default.xml` to verify that encryption is configured correctly, and that the password is no longer recorded in clear text.

```
<authenticationProvider controlFlag="optional"
name="com.sybase.security.core.PreConfiguredUserLoginModule">
<options name="username" value="supAdmin"/>
<options name="roles" value="SUP Administrator,SUP Domain
Administrator,SUP DCN User"/>
<options encrypted="true" name="password" value="1-
AAAAEgQQWd8NguXX5nswpWF1vUFPtcJhjmoiSYUzEAAiY3vWkZ+Y/33cWAoUD+EV/
D80Yo4vie/
XIyZVoBZbTT9ijxHDe7wbIBsagzS0DdAvS5lTRvRRNVp83+pTjQ3mmMNt5FmxrGvU
V5fVQ2JI1YaTPbd+Tw==">
```

See also

- *Enabling Authentication and RBAC for Administrator Logins* on page 31

Preparing SSL for HTTPS Listeners

SSL requires that two self-signed certificates installed by default need to be replaced: one for Unwired Server, and one for Sybase Control Center. Once the certificates have been imported, you can configure the security profile for these components in Sybase Control Center.

1. *Determining Certificate Requirements Based on Security Profile Chosen*

   By default, Unwired Server includes two security profiles, which are used by secure management of Unwired Server from Sybase Control Center and Data Change Notification (DCN) listeners: default and default_mutual.

2. *Changing Installed Certificates Used for Unwired Server and Sybase Control Center HTTPS Listeners*

   Both Unwired Server and Sybase Control Center include default certificates that are used for these components' HTTPS listeners. Since all installations use the same certificates by default, you must change these certificates with production-ready ones after you install Unwired Platform. Unwired Server and Sybase Control Center share the same keystore and truststore (that is, SUP_HOME\Servers\UnwiredServer\Repository\Security\).

3. *Enabling and Configuring Administration Encryption for Unwired Server*

   Enable encryption to securely transfer data between the Unwired Server administration listener and Sybase Control Center.

See also

- *Securing Multiple Domains* on page 47

Determining Certificate Requirements Based on Security Profile Chosen

By default, Unwired Server includes two security profiles, which are used by secure management of Unwired Server from Sybase Control Center and Data Change Notification (DCN) listeners: default and default_mutual.

The security profile you use determines which certificate file you need, and where they need to be deployed. The most secure profile is default_mutual, whereby components are mutually authenticated.

For details about what cipher suites are supported for domestic and domestic_mutual authentication, see *Creating an SSL Security Profile in Sybase Control Center* in the *Sybase Control Center for Sybase Unwired Platform*.

1. The default security profile uses domestic authentication. With this authentication type, Unwired Server sends its certificate to the client (that is, either Sybase Control Center or DCNs). However, it does not require a certificate in return from the client. If you choose this option then you need to:

   - Use the alias of "sample1".
   - Configure the Sybase Control Center to trust the Unwired Server certificate.

2. The default_mutual security profile uses domestic_mutual authentication. If you use this option then you need to:
   - Use the alias of "sample2".
   - Ensure both Sybase Control Center and Unwired Server truststores each contain a copy of the other component's certificate.

Changing Installed Certificates Used for Unwired Server and Sybase Control Center HTTPS Listeners

Both Unwired Server and Sybase Control Center include default certificates that are used for these components' HTTPS listeners. Since all installations use the same certificates by default, you must change these certificates with production-ready ones after you install Unwired Platform. Unwired Server and Sybase Control Center share the same keystore and truststore (that is, *SUP_HOME*\Servers\UnwiredServer\Repository\Security\).

To share certificates, Sybase recommends that you maintain the existing certificate alias (that is, "sample1" or "sample2" depending on the profile used) in the new certificates. Then, when you replace the IIOPS default certificate with the new production certificate, you are updating change the certificate for all listeners simultaneously.

Note: Because secure DCN has automatically been configured to use these same profiles by default, you are updating certificates used for secure DCN communication. If you want DCN to use a unique profile and certificates, see *Creating a Unique SSL Profile For DCN*.

1. Generate new production-ready certificates:
   a) Use your PKI system to generate Unwired Server certificates and key pairs, and have them signed with the Certificate Authority (CA) certificate used in your organization.

      Ensure that you:
      - Keep the required alias for your profile type.
      - Set the CN of the certificate to *.*MyDomain*. The truststore and keystore files, as well as the definitions for default and default_mutual profiles are then synchronized across the cluster. As a result, there will only ever be a single certificate shared by all nodes that are members of the same cluster.

      Unwired Platform is compliant with certificates and key pairs generated from most well known PKI systems.
   b) For Sybase Control Center: generate a new certificate with a "jetty" alias. This replaces the default self-signed certificate installed for this component specifically.
2. Import production-ready certificates, then update the security profile to associate these files with the Unwired Server encrypted port.
   a) Use **keytool** to import the new production certificates into the primary Unwired Server keystore.
   b) In the left navigation pane, select **Configuration**.

c) In the right administration pane, click **General** then **SSL Configuration**.
d) Optional. If you have used a different alias, rather than keep the alias of "sample1", locate the profile name row and modify the alias name to match the one used by your certificate.
e) Optional. If you are using a PKI system that includes OCSP, configure an OCSP responder. See *Enabling OCSP*.

3. Replace the default certificate for Sybase Control Center's HTTPS listener. Use **keytool** to import the new Sybase Control Center certificate with the "jetty" alias to the *SCC_HOME* `\keystore` keystore.

See also

- *Enabling and Configuring Administration Encryption for Unwired Server* on page 17
- *Unwired Server and Sybase Control Center Communications* on page 6

Enabling OCSP

(Optional) Enable OCSP (Online Certificate Status Protocol) to determine the status of a certificate used to authenticate a subject: current, expired, or unknown. OCSP configuration is enabled as part of cluster level SSL configuration. OCSP checking must be enabled if you are using the CertificateAuthenticationLoginModule and have set Enable revocation checking to true.

Enable OCSP for a cluster when configuring SSL.

1. In the left navigation pane, select **Configuration**.
2. In the right administration pane, select the **General** tab.
3. From the menu bar, select **SSL Configuration**.
4. To enable OCSP when doing certificate revocation checking, check **Enable OCSP**.
5. Configure the responder properties (location and certificate information):

Responder Property	Details
URL	A URL to responder, including its port. For example, `https://ocsp.example.net:80`.

Responder Property	Details
Certificate subject name	The subject name of the responder's certificate. By default, the certificate of the OCSP responder is that of the issuer of the certificate being validated. Its value is a string distinguished name (defined in RFC 2253), which identifies a certificate in the set of certificates supplied during cert path validation. If the subject name alone is not sufficient to uniquely identify the certificate, the subject value and serial number properties must be used instead. When the certificate subject name is set, the certificate issuer name and certificate serial number are ignored. For example, `CN=MyEnterprise, O=XYZCorp`.
Certificate issuer name	The issuer name of the responder certificate. For example, `CN=OCSP Responder, O=XYZCorp`.
Certificate serial number	The serial number of the responder certificate.

See also

- *Creating an SSL Security Profile in Sybase Control Center* on page 108

Using Keytool to Generate Self-Signed Certificates and Keys

Whenever possible, use a PKI system and a trusted CA to generate production-ready certificates and keys that encrypt communication among different Unwired Platform components. You can then use **keytool** to import and export certificate to the platform's keystores and truststores. Otherwise, you can also use **keytool** to generate self-signed certificates and keys.

Review sample commands, to see how to use **keytool** to import, export, and generate certificates and keys. For more information, see *Configuring X.509 Certificates for SAP Single Sign-On*.

1. If you have the root certificate of the certificate authority (CA) or if you have a self-signed certificate, import the CA certificate into the keystore and truststore.
   For example, if you have a CA certificate in a PKCS#10 file named `cust-ca.crt`, run this command from the *SUP_HOME*`\Servers\UnwiredServer\Repository\Security` directory:

```
keytool -importcert -alias customerCA -file cust-ca.crt -storepass
changeit -keystore truststore.jks -trustcacerts
```

The truststore is used when Unwired Platform makes an out-bound connection over SSL to another server with a server certificate. Unwired Server checks that the server certificate is in the truststore, or is signed by a CA certificate in the truststore.

2. Generate a key pair in the Unwired Platform keystore.

   The command you use depends on the environment for which you are generating the keystore. For most Unwired Platform deployments, this command may be sufficient:

   ```
   keytool -genkeypair -alias supServer -keystore keystore.jks -
   keyalg RSA -keysize 2048
   -validity 365 -keypass mySecret -storepass changeit
   ```

   However, if you are generating a key pair to secure an HTTPS communication port between the SAP Gateway and Unwired Server for OData push notifications, you might use a command like:

   ```
   keytool –genkeypair –alias SAPpush –keyalg RSA –keysize 1024 –
   sigalg SHA1withRSA
   –keypass mySecret –keystore keystore.jks
   ```

3. Supply values for each of the resulting prompts.

   The first prompt is the most critical. If you are running multiple Unwired Server in a cluster, type an asterisk followed by the domain name where the Unwired Servers are running.

   ```
   What is your first and last name?
   [Unknown]: *.mydomain.com
   What is the name of your organizational unit?
   [Unknown]: myOU
   What is the name of your organization?
   [Unknown]: mycompany
   What is the name of your City or Locality?
   [Unknown]: place
   What is the name of your State or Province?
   [Unknown]: state
   What is the two-letter country code for this unit?
   [Unknown]: AB
   Is CN=*.mySUPdomain.com, OU=myOU, O=mycompany,
   L=place, ST=state, C=AB correct?
   [no]: y
   ```

   Note: The asterisk before the domain name allows this same certificate to be used by multiple Unwired Servers deployed as members of a common cluster. The CN value must be the domain name of the host on which Unwired Server is installed.

4. Generate a certificate signing request, send it to the certificate authority, and install the issued certificate in the Unwired Server keystore:

   a) Generate a certificate signing request (CSR). For example:

   ```
   keytool -certreq -alias supServer -keystore keystore.jks -
   storepass changeit
   ```

```
-keypass mySecret -file supServer.csr
```

b) Send the CSR to the CA for signing.
For example, for SAP, may perform steps similar to:
   1. Launch the URL for your SAP CA.
   2. Change the option to **Certify the cert req** in the **select cmd** option.
   3. Paste the content of the `.csr` file generated in the previous step.
   4. Copy the content between (and including) `"-----BEGIN CERTIFICATE-----" "-----END CERTIFICATE-----"` of the response, to a text file named *<name of the cert>*`.cer`.
   5. View and verify the status of the certificate.

c) Use `keytool` to import the CA.

Note: The -alias/-keypass values are the same as those used to generate the key pair and CSR. By sharing these values, you pair the signed certificate with the keypair:

```
keytool -importcert -alias supServer -file supServer.crt -
keypass mySecret -storepass changeit
-keystore keystore.jks -trustcacerts
Certificate reply was installed in keystore
```

See also

- *Certificate Authentication Properties* on page 164

Enabling and Configuring Administration Encryption for Unwired Server

Enable encryption to securely transfer data between the Unwired Server administration listener and Sybase Control Center.

You can create or change a security profile that saves SSL setup data for a particular server instance. Using the security profile, you associate a specific key with the encrypted port.

1. In the left navigation pane, expand the **Servers** folder and select a server.
2. Select **Server Configuration**.
3. In the right administration pane, click **General**.
4. Optional. If you want to create a new security profile, select **SSL Configuration**.
5. In the **Configure security profile table**:
   a) Enter a name for the security profile.
   b) Enter a certificate alias. This is the alias of a key entry in the keystore. Make sure the key password of this key entry is the same as the keystore password.
   c) Select an authentication level:

      If the security profile authenticates only the server, then only the server must provide a certificate to be accepted or rejected by the client. If the security profile authenticates both the client and the server, then the client is also required to authenticate using a

certificate; both the client and server will provide a digital certificate to be accepted or rejected by the other.

Profile	Authenticates	Cipher suite(s)
intl	server	• SA_EXPORT_WITH_RC4_40_MD5 • RSA_EXPORT_WITH_DES40_CBC_SHA
intl_mutual	client/server	• RSA_EXPORT_WITH_RC4_40_MD5 • RSA_EXPORT_WITH_DES40_CBC_SHA
simple	server	RSA_WITH_NULL_MD5 RSA_WITH_NULL_SHA
simple_mutual	client/server	RSA_WITH_NULL_MD5 RSA_WITH_NULL_SHA
strong	server	• RSA_WITH_3DES_EDE_CBC_SHA • RSA_WITH_RC4_128_MD5 • RSA_WITH_RC4_128_SHA
strong_mutual	client/server For example, this is the required option for mutual authentication of Unwired Platform and Gateway.	• RSA_WITH_3DES_EDE_CBC_SHA • RSA_WITH_RC4_128_MD5 • RSA_WITH_RC4_128_SHA
domestic	server	• RSA_WITH_3DES_EDE_CBC_SHA • RSA_WITH_RC4_128_MD5 • RSA_WITH_RC4_128_SHA • RSA_WITH_DES_CBC_SHA • RSA_EXPORT_WITH_RC4_40_MD5 • RSA_EXPORT_WITH_DES40_CBC_SHA • TLS_RSA_WITH_NULL_MD5 • TLS_RSA_WITH_NULL_SHA

Profile	Authenticates	Cipher suite(s)
domestic_mutual	client/server	• RSA_WITH_3DES_EDE_CBC_SHA • RSA_WITH_RC4_128_MD5 • RSA_WITH_RC4_128_SHA • RSA_WITH_DES_CBC_SHA • RSA_EXPORT_WITH_RC4_40_MD5 • RSA_EXPORT_WITH_DES40_CBC_SHA • RSA_WITH_NULL_MD5 • RSA_WITH_NULL_SHA

6. Use IIOPS in the Communication Ports sub-tab by selecting **Secure Management Port** (port 2001), and ensure that Sybase Control Center's Managed Resource properties match. By default, IIOPS is already configured between Unwired Server and Sybase Control Center.
7. Select the correct security profile name that provides the details for locating the correct certificates.
8. Save the changes and restart the server.

See also

• *Changing Installed Certificates Used for Unwired Server and Sybase Control Center HTTPS Listeners* on page 41

Securing Multiple Domains

To prevent role mapping leaks between multiple tenant domains, configure domains and assign shared security configurations.

Sybase recommends that the Platform administrator:

1. Create at least one new tenant domain in Sybase Control Center. You may require more, depending on your mobility strategy.
2. Restrict the use of the "admin" security configuration on the "default" domain to administration authentication only.
3. Assign at least one domain administrator. Depending on the maintenance issues of large-scale deployments, the administrator may want to use at least one Domain administrator per domain.
4. Create and assign at least one new security configuration. The administrator may create and assign security configurations, if security requirements (stringency, uniqueness) differ between tenant domains.

For more information, search for *Domains* in *Sybase Control Center for Sybase Unwired Platform*.

For example, a company named "Acme" has two separate divisions, HR and sales. The employees in each division use different mobile applications. In this case, Sybase recommends using two domains in Sybase Control Center to simplify the management of packages, users, applications and related artifacts.

Acme implements separate domain administrators for each domain, but is using a single "acme" security configuration due to the way the corporate LDAP directory is configured. This configuration includes an LDAPLoginModule provider that uses this URL:

```
ldap://ldap.acme.com
```

As a result, all employees of all domains are authenticated by the same LDAP server, and authorized by the same set of groups and roles.

Note: Because domain administrators are authenticated from the same acme LDAP repository via the admin security configuration on the default domain, those role mappings can "leak" between domains. Consequently, a domain administrator assigned to one domain gets granted access to another. This side-effect is undesirable and should be avoided.

See also

- *Preparing SSL for HTTPS Listeners* on page 40
- *Configuring Unwired Server to Securely Communicate With an HTTP Proxy* on page 50

Determining a Tenancy Strategy

Determine how many domains to create and how to distribute domain components. A strategic multitenant structure for the cluster balances system-availability considerations with administrative concerns.

Domains are primarly containers for packages for a specific group. This group is called a tenant and can be internal to a single organization or external (for example, a hosted mobility environment).

Packages are attached to named security configurations that determine which users have access to mobile business object data, so you must create at least one security configuration and assign it to the domain for which the package is being deployed. You must identify which users require access to each package and how you will distribute the packages across the system using domains to logically partition the environment.

1. Organize device users according to the data they need to access. Ideally, create a domain for each distinct set of users who access the same applications and authenticate against the same back-end security systems. If you do not need to support multiple groups in distinct partitions, then the single default domain should suffice.
2. Consider how these groups will be affected by administrative operations that prevent them from synchronizing data. Sometimes, you can limit the number of users affected by

administration and maintenance disruptions by distributing packages across additional domains. Operationally, the more components a domain contains, the more clients who are unable to access package data during administrative operations like domain synchronizations.

3. Assess the administrative resources of the tenant to determine how much time can be committed to domain administration tasks. Certain multitenant configurations require a greater amount of administrative time. For example, if you distribute packages from the same EIS across a number of domains, you must maintain identical data source configurations for each of these packages. This option requires more administrative time than grouping all packages belonging to the same EIS into one domain.
4. Decide how many domains to create for the customer, and identify which packages to group within each domain, according to the needs of the user groups you identified in step 1.

Benefits and Drawbacks of a Shared Security Configuration

Determine whether or not to use a shared security configuration across multiple domains.

Sybase recommends that you use differently named security configurations for each domain, unless you are willing to accept the risks, and domain administrators collaborate before implementing changes.

Benefit	Drawback
• Set up the modules you required in a named security configuration once.	• A domain administrator from one domain can make changes to role mapping at the default level, potentially with adverse effects to packages deployed to a different domain.

Creating and Enabling a New Domain

Create and configure multiple domains within a single Unwired Platform installation. A domain must be enabled for application users to access the packages deployed in the domain. Enabling a domain also triggers synchronization of the domain changes to the secondary nodes in the cluster. Application users who attempt to access a disabled domain receive an error message.

Prerequisites

Create a security configuration for the domain and register the domain administrator.

Task

1. Open Sybase Control Center.
2. In the left navigation pane, select the **Domains** folder.
3. In the right administration pane, select the **General** tab, and click **New**.

4. In the Create Domain dialog, enter a name for the domain and click **Next**.
5. Select a security configuration for the domain by checking an option from the list of available configurations. You must select at least one security configuration. The security configurations you select are then available for use in validating users accessing the packages. If you select multiple security configurations, the first one you select becomes the default security configuration for the domain.
6. Click **Next**.
7. Optional. Select one or more domain administrators for the domain.
8. Click **Finish**.
   The new domain appears in the **General** tab.
9. Click the box adjacent to the domain name, click **Enable**, then click **Yes** to confirm.

Configuring Unwired Server to Securely Communicate With an HTTP Proxy

If you want Unwired Server connect to an HTTP Proxy, you can set connection properties for it when you optimize Unwired Server performance.

1. Open Sybase Control Center.
2. In the left navigation pane, expand the **Servers** folder and select a server.
3. Select **Server Configuration**.
4. In the right administration pane, select the **General** tab.
5. In the User Options row, enter command-line options to control the startup behavior.
   Enter options as a series of Java system property and value pairs, using this syntax:
   ```
   -DpropertyName=value
   ```
   Supported properties include:
   - **`-Dhttp.proxyHost`** – for the host name of the proxy server.
   - **`-Dhttp.proxyPort`** – for the port number. The default value is 80.
   - **`-Dhttp.nonProxyHosts`** – for the list of hosts that should be reached directly, thereby bypassing the proxy. Separate multiple entries with `|`.

     The patterns may start or end with a `*` for wildcards. A host name that matches the wildcard pattern can bypass the proxy. For example, to use command-line options to configure a proxy and other non-proxy hosts (including those on local computers):
     ```
     -Dhttp.proxyHost=proxy.myDomain.com -Dhttp.proxyPort=8080 -
     Dhttp.nonProxyHosts=*.myOtherDomain1.com|localhost|
     *.myOtherDomain2.corp
     ```
6. Click **Save**.
7. Restart the server for changes to take effect.

See also

- *Securing Multiple Domains* on page 47

Enabling Authentication and RBAC for User Logins

Enable authentication and role-based access control (RBAC) for device user logins by creating a new security configuration (that is, one that is distinct from the "admin" security configuration on the "default" domain), and mapping roles, then assigning it.

Of all the default roles included with Unwired Platform, only the "SUP DCN User" role must be mapped, and only if DCNs are used for MBO packages associated with the security configuration you create for device user logins.

See also

- *Securing Platform Administration* on page 30
- *Encrypting Synchronization for Replication Payloads* on page 101
- *Authentication in Unwired Platform* on page 52

Supported Providers and Credential Types

Different security providers allow users to supply different user credentials. If your security policy mandates that a specific credential type or strength be used, review the providers that are available to you.

Table 1. Credentials and Providers

Credential	Providers Available
User name and password	LDAP, NTProxy, HTTP
E-mail address and password. Note that E-mail addresses must follow certain requirements for Sybase Unwired Platform to recognize them correctly, especially when a security configuration is defined with it. See *Considerations for Using E-mail Addresses as User Names*.	LDAP, HTTP
X.509 certificates	Certificate, SAP SSO
Tokens	HTTP

Considerations for Using E-mail Addresses as User Names

At registration, application users can use an e-mail address as a user name in Unwired Platform. However, those users must ensure that e-mail addresses are processed correctly, especially when a security configuration is paired with the e-mail address.

A valid e-mail address:

- Can use any combination of uppercase and lowercase English alphanumeric characters (a–z, A–Z, and 0–9)
- Is limited to:
  - These special characters, which you can use without escape characters: .!#$%&'*+-/=?^_`{|}~ (that is, ASCII 33, 35–39, 42, 43, 45, 46, 47, 61, 63, 94–96, and 123–126)
  - These special characters, with which you must use an escape character: "(),:;<>@[\] (that is, ASCII 32, 34, 40, 41, 44, 58, 59, 60, 62, 64, and 91–93)
- For Unwired Platform 2.1 ESD #3, user name cannot exceed 100 characters for messaging applications or 128 characters for other application types.
- The user name length limit for packages deployed on earlier versions of Unwired Platform is still 36.

Note: This syntax information is only for your reference; while Unwired Server validates strings to ensure there are no restricted characters, it does not validate addresses to ensure they are syntactically correct.

When you use an e-mail address as the user name, ensure that the e-mail address domain is followed with a "." to prevent the address from being misinterpreted as a security configuration name. For example, jdoe@domain.com.

Table 2. E-mail Address Parsing Examples

User Name Entered	Result
userID	A user ID string. No risk of misinterpretation.
userID@textA	The user is authenticated with the "textA" security configuration if it exists.
userID@textb.com	An e-mail address as user name. No security configuration identified.

Authentication in Unwired Platform

A security provider verifies the identities of application users and administrators who request access via one or more configured login modules.

Device user authentication and administrator authentication are configured differently:

- device users are authenticated with custom Unwired Server security configurations created by the platform administrator in Sybase Control Center. For SAP EIS backends, SSO authentication can be configured.
- Administrators are authenticated with the "admin" security configuration on the "default" domain. For first-time logins, administrators are authenticated with the PreconfiguredLoginModule. Once logged in, administrators for production systems should immediately reconfigure security to use the enterprise security backend and delete this login module from Sybase Control Center.

Caching Authenticated Sessions

An authentication request with username/password or certificate credentials for a specific domain always results in looking up an existing authenticated session in the cache that used the same credentials. If one is found, the session is reused instead of delegating the authentication request to the configured security backend. This is the case even if any of the information from the client session is used to authenticate the user instead of the presented username/password or certificate credentials.

If an existing authenticated session is found in the cache with the same credentials, then the user is not authenticated again against the configured security backend even if the cached session was authenticated based on an http header/cookie/personalization value and the new authentication request contains a different value for that parameter.

See also

- *Enabling Authentication and RBAC for User Logins* on page 51
- *Enabling Authentication and RBAC for Administrator Logins* on page 31

Creating a Security Configuration for Device Users

Create and name a set of security providers and physical security roles to protect Unwired Platform resources. For device user authentication, create at least one security configuration that is not the "admin" security configuration on the "default" domain, which is used exclusively for administrator authentication in Sybase Control Center.

Only platform administrators can create security configurations. Domain administrators can view configurations only after the platform administrator creates and assigns them to a domain.

1. In the left navigation pane, expand the **Security** folder.
2. In the right administration pane, click **New**.
3. Enter a name for the security configuration and click **OK**.

Assigning Providers to a Security Configuration

Assign providers after you have created a security configuration.

1. In the left navigation pane, expand the **Security** folder.
2. Select the security configuration you want to assign a provider to.
3. In the right administration pane, select the **Settings** tab to set an authentication cache timeout value.

   The timeout determines how long authentication results should be cached before a user is required to reauthenticate. For details, see *Authentication Cache Timeouts*. To configure this value:

   a) Set the cache timeout value in seconds. The default is 3600.
   b) Click **Save**.

4. Select the tab corresponding to the type of security provider you want to configure: Authentication, Authorization, Attribution, or Audit.
5. To edit the properties of a preexisting security provider in the configuration:
   a) Select the provider, and click **Properties**.
   b) Configure the properties associated with the provider by setting values according to your security requirements. Add properties as documented in the individual reference topics for each provider.
   c) Click **Save**.
6. To add a new security provider to the configuration:
   a) Click **New**.
   b) Select the provider you want to add.
   c) Configure the properties associated with the provider by setting values according to your security requirements. Add properties as documented in the individual reference topics for each provider.
   d) Click **OK**.
   The configuration is saved locally, but not yet committed to the server.
7. Select the **General** tab, and click **Validate** to confirm that Unwired Server accepts the new security configuration.
8. Click **Apply** to save changes to the security configuration, and apply them across Unwired Server.

Next

- If you have multiple providers, understand how to stack or sequence them and know what the implication of provider order means.
- Be sure to remove the NoSecurity providers from the Authentication, Authorization, and Attributer tabs. For more information, see *NoSecurity Provider*.

Authentication Cache Timeouts

Set a cache timeout value to cache user or administrator authentication credentials, which improves runtime performance.

Set timeout properties to avoid repeatedly reauthenticating users—a benefit of particular interest for device clients that receive separately authenticated messages. If a user logs in successfully, he or she can reauthenticate with the same credentials without validating them against a security repository. However, if the user provides a user name or password that is different from the ones cached, Unwired Server delegates the authentication request to the security repository.

This property affects only authentication results:

- The successful authentication result is cached.
- If authentication passes, user roles are assigned.

Authorization results and failed authentication results are not cached.

For example, if an MBO is protected by "LogicalRoleA", and the security configuration that MBO is deployed in has a role mapping to "PhysicalRoleA", each time a user tries to access this MBO, the provider checks to see if they are in PhysicalRoleA based on cached role membership from the original authentication. It does not check the security repository each time thereafter.

By default, the cache timeout value is 3600 (seconds). This value is enabled whether the property exists or not. You can change this value by configuring a new value for the property in Sybase Control Center for the appropriate security configuration. Or, you can set the value to 0, to restrict access and force reauthentication.

Enabling CRLs

Identify the certificate revocation lists (CRLs) that define a list of digital certificates which have been revoked. Revoked certificates should not give the Sybase Unwired Platform device user access to the Unwired Server runtime.

Administrators can configure certificate revocation lists (CRLs) to check if any of the certificates in the path are revoked. A series of URIs define the CRL location.

1. Using Sybase Control Center, open the CertificateAuthenticationLoginModule and CertificateValidationLoginModule used by your security configuration.
2. For the CRL property, define one or more URIs. If using multiple URIs, each must be indexed.

   The index number used determines the order in which CLRs are checked. This example uses two URI, each indexed accordingly so that the Verisign CRL comes first.

   ```
   crl.1.uri=http://crl.verisign.com/
   ThawtePersonalFreemailIssuingCA.crl
   crl.2.uri=http://crl-server/
   ```

Next

Note: While CRL applies to a particular login module, OCSP determines certificate status server-wide. Administrators must edit the `%JAVA_HOME%/jre/security/java.security` file to enable OCSP. Then in the login modules, set the Enable Revocation Checking property to true. For information, see *Enabling OCSP*.

Built-in Security Providers for User Authentication and Authorization

Unwired Server supports a variety of built-in security providers use to authenticate device users. Administrators create a security configuration and assign one or more providers to it using Sybase Control Center.

You can configure a provider of a given type only if that provider is available on the enterprise network.

If you are using Unwired Server in an Online Data Proxy deployment, not all providers are applicable in this environment.

NoSecurity Provider

The NoSecurity provider offers pass-through security for Unwired Server, and is intended for use in development environments or for deployments that require no security control. Do not use this provider in production environments — either for administration or device user authentication.

If you use the NoSecurity provider, all login attempts succeed, no matter what values are used for the user name and password. Additionally, all role and control checks based on attributes also succeed.

Sybase provides these classes to implement the NoSec provider:

- **NoSecLoginModule** – provides pass-through authentication services.
- **NoSecAttributer** – provides pass-through attribution services.
- **NoSecAuthorizer** – provides pass-through authorization services.

For more information, see *NoSecurity Configuration Properties*.

LDAP Security Provider

The LDAP security provider suite includes authentication, attribution, and authorization providers. Add an LDAP provider to a security configuration to authenticate administrator logins (on the "admin" security configuration on the "default" domain) or device user logins (any custom security configuration for that purpose).

You can configure these providers:

- The **LDAPLoginModule** provides authentication services. Through appropriate configuration, you can enable certificate authentication in **LDAPLoginModule**.
- (Optional) **LDAPAuthorizer** or **RoleCheckAuthorizer** provide authorization service in conjunction with LDAPLoginModule. LDAPLoginModule works with either authorizer.

  The **RoleCheckAuthorizer** is part of every security configuration but does not appear in Sybase Control Center.

  Use **LDAPAuthorizer** only when **LDAPLoginModule** is not used to perform authentication, but roles are still required to perform authorization checks against the LDAP data store. If you use **LDAPAuthorizer**, always explicitly configure properties; for it cannot share the configuration options specified for the **LDAPLoginModule**.
- (Optional) **LDAPAttributer** is used to retrieve the list of roles from the LDAP repository. These roles are displayed in the role mapping screen in Sybase Control Center. The LDAP attributer is capable of sharing the configuration properties from the LDAPLoginModules. If no configuration properties are explicitly specified, then the attributer iterates through the configured LDAPLoginModules and retrieves the roles from all the LDAP repositories configured for the different LDAPLoginModules.

You need not enable all LDAP providers. You can also implement some LDAP providers with providers of other types. If you are stacking multiple LDAP providers, be aware of and understand the configuration implications.

See also

- *LDAP Configuration Properties* on page 153
- *Adding a Production-Grade Provider* on page 33
- *Stacking Providers and Combining Authentication Results* on page 97

Configuration Best Practices for Multiple LDAP Trees

Use the Sybase Unwired Platform administration perspective to configure LDAP authentication and authorization security providers, which are used to locate LDAP user information when organizational user groups exist within multiple LDAP trees.

To accommodate an LDAP tree structure that cannot be directly accessed using one search base:

- Create an LDAP authentication module for each level in the hierarchy – during the authentication process, Unwired Platform tries to authenticate against every login module in the ordered list until authentication succeeds or until it reaches the end of the list. Depending on the number of login modules you configure, this approach may have some performance issues.
- Use different AuthenticationScopes for performing user searches – specify the root node of a particular LDAP tree, by entering `AuthenticationSearchBase="dc=sybase, dc=com"` and `set Scope=subtree`. Unwired Platform performs an LDAP query against the entire subtree for authentication and authorization information. Depending on the number of AuthenticationScope within the LDAP tree structure, this approach can have performance implications.
- If multiple servers are clustered together to form a large logical directory tree, configure the LDAPLoginModule by setting the `Referral` property to `follow`.
- If subjects have been made members of too many LDAP groups and the search for physical roles results in too many results, the maximum result limit may be reached and authentication fails. To avoid this, narrow the RoleSearchBase to LDAP groups that are relevant only to Sybase Unwired Platform. Sybase also recommends setting the **SkipRoleLookup** property to true to eliminate the need to search all the roles defined in the role search base.

LDAP Role Computation

Role checks are the primary means of performing access control when using LDAP authentication. Authentication and attribution capabilities both utilize role computation techniques to enumerate roles that authenticated users have.

There are three distinct types of role constructs supported by LDAP providers; each may be used independently, or all three may be configured to be used at the same time.

- User-level role attributes, specified by the `UserRoleMembershipAttributes` configuration property, are the most efficient role definition format. A user's roles are enumerated by a read-only directory server-managed attribute on the user's LDAP record. The advantage to this technique is the efficiency with which role memberships can be

queried, and the ease of management using the native LDAP server's management tools. These constructs are supported directly by ActiveDirectory, and use these configuration options:

- UserRoleMembershipAttributes – the multivalued attribute on the user's LDAP record that lists the role DNs that the user is a member of. An example value for this property is "memberOf" on ActiveDirectory.
- RoleSearchBase – the search base under which all user roles are found, for example, "ou=Roles,dc=sybase,dc=com". This value may also be the root search base of the directory server.
- RoleFilter – the search filter that, coupled with the search base, retrieves all roles on the server.
- (Optional) RoleScope – enables role retrieval from subcontexts under the search base.
- (Optional) RoleNameAttribute – choose an attribute other than "cn" to define the name of roles.

These properties are set to default values based on the configured server type. However, these properties can be explicitly set to desired values if the server type is not configured or set to overwrite the default values defined for a server type.

- LDAP servers allow groups to be members of other groups, including nested groups. The LDAP provider does not compute the group membership information recursively. Instead, nested group membership information is taken into consideration for role computation only if the LDAP server provides a user attribute that contains the complete list of group memberships, including static, dynamic, and nested group memberships. See *LDAP Nested Groups and Roles in LDAP*.
- Freeform role definitions are unique in that the role itself does not have an actual entry in the LDAP database. A freeform role starts with the definition of one or more user-level attributes. When roles are calculated for a user, the collective values of the attributes (each of which may can be multivalued) are added as roles to which the user belongs. This technique may be useful when the administration of managing roles becomes complex. For example, assign a freeform role definition that is equivalent to the department number of the user. A role check performed on a specific department number is satisfied only by users who have the appropriate department number attribute value. The only property that is required or used for this role mapping technique is the comma-delimited `UserFreeformRoleMembershipAttributes` property.

LDAP Provider Stacking and Configuration Sharing

LDAP login and attribution modules can sometimes share a common configuration. LDAPAttributer can share the configuration properties from the configured LDAP login modules only if no configuration properties are explicitly configured for LDAPAttributer.

When stacking these modules, be aware that authorizers do not inherit configuration properties from the login modules you configure. Configurations must be explicit. In the case where both LDAPLoginModule and LDAPAuthorizer are separately configured in a :

- Matching configuration, then LDAPAuthorizer simply skips the role retrieval.

- Differing configuration, then LDAPAuthorizer proceeds with the role retrieval from the configured back-end, and performs the authorization checks using the complete list of roles (from both the login module and itself).

Only one attributer instance needs to be configured even when multiple login module instances are present in the security configuration. The LDAPAttributer attributes an authenticated subject using the LDAP configuration that was used to authenticate the subject. However, the list of available roles is computed by the LDAPAttributer by iterating through all available LDAP configurations.

When using LDAPAttributer stacking and configuration, keep in mind:

- LDAPAttributer has maximum functionality when combined with the LDAP authentication provider; the LDAPAttributer can be configured completely standalone or with alternate authentication providers.
- If you do not configure an LDAPLoginModule, you must define the configure all properties in the attributer.
- If explicit configuration properties are specified for the attributer, then the properties from the login module are not used for attributer functionality, including retrieving attributes for authenticated subjects, listing roles, and more. Sybase recommends that you share configurations rather than trying to maintain separate configurations.

Nested Groups and Roles in LDAP

The LDAP provider computes the roles granted to an authenticated user using the role and group membership information from the LDAP repository. To support nested roles and groups, LDAP servers allow roles and groups to be members of other roles and groups respectively.

The LDAP provider retrieves role membership from the user attribute specified by UserRoleMembershipAttributes configuration property, and does not compute the role membership information recursively. Therefore any nested and dynamic roles are taken into consideration only if the LDAP server provides a user attribute that contains the complete list of role memberships, including static, dynamic, and nested role memberships. For example, in SunOne server, the UserRoleMembershipAttributes property for the LDAP provider should be set to "nsRole" instead of the default value "nsRoleDN" to enable it to retrieve the nested roles information.

Similarly LDAP group memberships are stored and checked on a group-by-group basis. Each defined group, typically of objectclass `groupofnames` or `groupofuniquenames`, has an attribute listing all of the members of the group. The LDAP provider does not support nested or dynamic groups (groups that are populated with objects found by doing an LDAP search rather than static members). For example, it does not recursively compute all the groups to which the user has membership. Therefore any nested and dynamic groups are taken into consideration only if the LDAP server provides a user attribute that contains the complete list of group memberships, including static, dynamic, and nested group memberships. For example, in Active Directory server, the UserRoleMembershipAttributes property for the

LDAP provider should be set to "tokenGroups" to enable it to retrieve the nested group membership information.

For additional information, see *Skipping LDAP Role Lookups (SkipRoleLookup)*, and *LDAP Configuration Properties*.

Skipping LDAP Role Lookups (SkipRoleLookup)

When configuring an LDAP provider, use the **SkipRoleLookup** configuration option to grant the user attributes defined in the **UserRoleMembershipAttributes** property.

Setting **SkipRoleLookup** to true grants all the roles retrieved using the **UserRoleMembershipAttributes** property from the LDAP user entry. The user roles are not cross-referenced with the roles retrieved from the role search base using the role search filter.

This eliminates the need to look up all the roles defined in the role search base and match the role filter as roles are retrieved. If the list of roles granted to the authenticated user is to be restricted to the roles defined in the role search base, set **SkipRoleLookup** to false.

Configuring an LDAP Provider to use SSL

If your LDAP server uses a secure connection, and its SSL certificate is signed by a nonstandard certificate authority, for example it is self-signed, use the keytool utility (**keytool.exe**) to import the certificate into the truststore.

1. Run the following console command: `keytool.exe -import -keystore SUP_HOME\Servers\UnwiredServer\Repository\Security \truststore.jks -file <LDAP server cert file path> -alias ldapcert -storepass changeit`.
2. Restart Sybase Unwired Platform services.
3. Log in to Sybase Control Centerfor Sybase Unwired Platform.
4. In the navigation pane of Sybase Control Center, expand the Security folder and select the desired security configuration in which to add the LDAP provider.
5. In the administration pane, click the **Authentication** tab.
6. Add an LDAPLoginModule, configuring the ProviderURL, Security Protocol, ServerType, Bind DN, Bind Password, Search Base, and other properties determined by you and the LDAP administrator. Choose **one** of the two methods below to secure a connection to the LDAP server:
   a) Use `ldaps://` instead of `ldap://` in the **ProviderURL**.
   b) Use `ssl` in the **Security Protocol**.
7. In the **General** tab, select **Validate** then **Apply**.
8. Click **OK**.

NTProxy Security Provider

NTProxy — sometimes known as native Windows login — is an Unwired Server provider that integrates with existing Windows login security mechanisms.

If added to a particular security configuration, users or administrators can authenticate with their native Windows user name and password, which gives them access to roles that are based on their existing Windows memberships.

The NTProxy provider fulfills authentication services only with classes in `csi-nativeos.jar`; role-based access control and attribution are not directly supported. Groups are also not supported in NTProxy. Instead, group memberships are transformed into a role of the same name and can be mapped in Sybase Control Center.

SAP SSO Token Security Provider

The SAPSSOTokenLoginModule has been deprecated and will be removed in a future release. Use HttpAuthenticationLoginModule for SAP SSO2 token authentication.

Use HttpAuthenticationLoginModule for both JCo and DOE-C connections to the SAP system. Unwired Server does not provide authorization control or role mappings for user authorization; enforce any access control policies in the SAP system.

See also

- *SAP SSO Token Authentication Properties* on page 176
- *Certificate Authentication Properties* on page 164
- *HTTP Basic Authentication Properties* on page 169

Certificate Security Provider

Use the Unwired Server CertificateAuthenticationLoginModule authentication provider to implement SSO with an SAP enterprise information system (EIS) with X.509 certificates.

Authorization control and role mappings for user authorization for EIS back-ends are enforced in the EIS using access control policies, not Unwired Server. For information on adding a mappable physical role for certificate authentication, see *Creating and Assigning a Security Configuration That Uses X.509 Credentials*, *UserRoleAuthorizer Provider*, and *Certificate Authentication Properties*.

See also

- *SAP SSO Token Authentication Properties* on page 176
- *Certificate Authentication Properties* on page 164
- *HTTP Basic Authentication Properties* on page 169

HTTP Authentication Security Provider
This provider is required when registration is set to automatic. It can also be used to enable SSO into SAP servers in place of the deprecated SAPSSOTokenLoginModule.

The LoginModule validates standard username/password style credentials by passing them to a Web server. Configure the URL property to point to a Web server that challenges for basic authentication.

This provider is enhanced to authenticate the user by validating a token specified by the client by sending the configured client values to the HTTP backend in the specified format (header/cookie). Any parameter value, for example personalization parameter, http header, or http cookie can be specified in the ClientHttpValuesToSend property so that the provider can retrieve the value of the configured parameter(s) and pass them to the Web server in the format required by the SendClientHttpValuesAs configuration property.

For example, to extract the cookie "MyCookie" from the client session to Unwired Server and pass it along to the Web server as the cookie "testSSOCookie", set the properties ClientHttpValuesToSend to "MyCookie" and set SendClientHttpValuesAs to cookie:testSSOCookie.

Note: Note that if "ClientHttpValuesToSend" property is configured, the provider only attempts to authenticate the user using those values. It does not set the username/password credentials in the http session to the Web server. If the specified client values are not found in the client session to SUP or if the Web server fails to validate the specified token, then this provider fails the authentication unless the property "TryBasicAuthIf TokenAuthFails" is set to `true` to enable it to revert to passing the username/password credentials to respond to the BasicAuth challenge.

Best practice guidelines include:

- Using an HTTPS URL to avoid exposing credentials.
- If the Web server's certificate is not signed by a well known CA, import the CA certificate used to sign the Web server's certificate into the Unwired Server `truststore.jks`. The truststore is prepopulated with CA certificates from reputable CAs.
- If this Web server returns a cookie as part of successful authentication, set the `SSO Cookie Name` configuration property to the name of this cookie. Upon successful authentication, this login module places the cookie value into an `HttpSSOTokenCredential` object and attaches it to the `java.security.Subject` as a public credential.

  Note: The HTTP Basic login module is the module that can either be used for SSO tokens or HTTP basic without SSO. The sole condition being that the backend support HTTP Basic authentication.
- When using this module in lieu of the deprecated SAPSSOTokenLoginModule, the cookie name is typically "MYSAPSSO2".

For example, SiteMinder is often used in mobile deployments to protect existing Web-based applications. Existing users point their browser at a URL, and SiteMinder intercepts an unauthenticated session to challenge for credentials (Basic). When the authentication succeeds, it returns a SMSESSION cookie with a Base64-encoded value that can be used for SSO into other SiteMinder enabled systems.

See *HTTP Basic Authentication Properties*.

See also

- *SAP SSO Token Authentication Properties* on page 176
- *Certificate Authentication Properties* on page 164
- *HTTP Basic Authentication Properties* on page 169

UserRoleAuthorizer Provider

The UserRoleAuthorizer provider grants logical roles to specific users when the user's roles cannot be retrieved by the configured login module from the backend. This provider cannot be configured.

This provider, which has no configuration properties and does not appear in the list of authorizers that can be configured in Sybase Control Center, is part of all security configurations created or updated in Sybase Control Center. UserRoleAuthorizer simply implements the checkRole method to compare the physical role name passed in to the current user name.

This authorizer allows the role check for the role "user:"+userName to succeed. For example, with this authorization module enabled, a domain administrator can use Sybase Control Center to map DCNRole to "user:jsmith". The user who authenticates as jsmith is then added in the physical role user:jsmith and is granted the logical DCNRole and can perform DCN.

CertificateValidationLoginModule Provider

Use a CertificateValidationLoginModule for mutual authentication.

Before any event is submitted to the Sybase Unwired Platform Runtime, the certificate must be validated. Additionally, the corresponding user name retrieved from the certificate, which is mapped to the logical role, must be authorized.

The CertificateValidationLoginModule can be used in conjunction with other login modules that support certificate authentication by configuring CertificateValidationLoginModule before configuring the login modules that support certificate authentication. You can only use this provider to validate client certificates when an HTTPS listener is configured to use mutual authentication.

For more information, see *Certification Validation Properties*, *UserRoleAuthorizer Provider*, and *Adding a certificateValidationLoginModule for DCN Mutual Authentication*.

Assigning Security Configurations to Domains, Packages, or Applications

A security configuration can be assigned to a domain. Domain administrators can then select the security configuration when deploying synchronization packages.

By selecting a security configuration at the package or application connection template level, you can choose the granularity required for user authentication. For details on how to assign and select a security configuration at the domain, package, or application level, search for *Security Configurations* in the *Sybase Control Center for Sybase Unwired Platform*.

Mapping Roles at the Global or Package Level

Unwired Platform uses a role mapper to map logical and physical roles during an access control check.

Role mappings can occur at two levels. Global role mappings are applied to all domains that are assigned the security configuration that contains the mappings. If you need role mappings to be package-specific, you can perform the mapping at the package level. Package-level mappings override the mappings set at the global level. Package-level role mappings apply to all packages that use the same security configuration, even if the package is deployed in multiple domains.

Administrators use logical roles to control access or group a large number of users who use the same security configuration. Administrators create and map the required logical roles and assign both a security configuration and a logical role to an application (through the application connection template). Users must have one of the physical roles that the logical role is mapped to in order to access the application.

Mapping State Reference

The mapping state determines the authorization behavior for a logical name instance.

State	Description
AUTO	Map the logical role to a physical role of the same name. The logical role and the physical role must match, otherwise, authorization fails.
NONE	Disable the logical role, which means that the logical role is not authorized. This mapping state prohibits anyone from accessing the resource (MBO or Operation). Carefully consider potential consequences before using this option.
MAPPED	A state that is applied after you have actively mapped the logical role to one or more physical roles. Click the cell adjacent to the logical role name and scroll to the bottom of the list to see the list of mapped physical roles.

Dynamically Mapping Physical Roles to Logical Roles

Map roles at either a domain or package level, depending on the scope requirements of a particular binding. If you use a particular role mapping for a package and a different role mapping at the domain level, the package mapping overrides the domain-level mapping.

In Sybase Control Center for Unwired Platform, determine where the role mapping needs to be applied:

- For domain-level mappings, configure role mappings as part of the security configuration for a domain. For details, see *Configuring Domain Security* in *Sybase Control Center for Sybase Unwired Platform*.
- For package-level mappings, configure role mappings when you deploy a package to Unwired Server, or at the package-level after deployment. For details, see *Assigning Package-Level Security* in *Sybase Control Center for Sybase Unwired Platform*.

SiteMinder Authentication with Sybase Unwired Platform

Configure your SiteMinder environment for authentication in Sybase Unwired Platform.

CA SiteMinder enables policy-based authentication and single sign-on with Sybase Unwired Platform. You can configure SiteMinder and Sybase Unwired Platform integration in a number of ways, depending on your environment. For detailed examples focusing on SiteMinder-specific configurations for Sybase Unwired Platform, see *How-To: Set up SUP with SiteMinder* at *http://scn.sap.com/docs/DOC-29574*.

SiteMinder Client Authentication

SiteMinder provides various client authentication options for Sybase Unwired Platform, including single sign-on (SSO), tokens, and Network Edge.

SiteMinder client authentication includes:

- Network Edge – when a reverse proxy or Relay Server in the DMZ is protected by SiteMinder, the Sybase Unwired Platform client is challenged for basic authentication credentials. If the credentials are valid, an SMSESSION cookie is issued and the client is allowed through to the Sybase Unwired Platform server. The client begins a session (RBS, MBS, or OData) by sending an HTTP(S) request to the reverse proxy. The reverse proxy detects the unauthenticated request, and challenges using basic authentication. After the 401 challenge, the client may already have network credentials configured, or executes a callback to prompt for credentials.
- Non-Network Edge – the Network Edge (reverse proxy or Relay Server) is not protected. The client's request is allowed to flow to Sybase Unwired Platform, where a LoginModule presents the basic credentials to a SiteMinder-protected Web server on behalf of the client. Sybase Unwired Platform server retains the SMSESSION cookie and credentials for the client.
- External tokens – the Sybase Unwired Platform client application obtains an SMSESSION cookie external to the Sybase Unwired Platform libraries using custom

application processing. This SMSESSION token passes into the Sybase Unwired Platform libraries as a cookie. Sybase Unwired Platform libraries add the cookie to subsequent HTTP requests to Sybase Unwired Platform server. The cookie may or may not be checked at the Network Edge.

- SAP SSO2 integration – the Sybase Unwired Platform user is initially authenticated by SiteMinder, resulting in an SMSESSION for the user. This SMSESSION is forwarded along with the SAP user ID to a SiteMinder SAP agent running inside of NetWeaver as a LoginModule. The SMSESSION is revalidated, and the TokenIssuingLoginModule is allowed to issue an SSO2 ticket for the specified SAP user ID. This ticket returns to Sybase Unwired Platform as an MYSAPSSO2 cookie. Sybase Unwired Platform now has both an SMSESSION and an SSO2 ticket to use for SSO purposes with various EIS depending on which SSO mechanism the EIS requires.

Note: In any of these authentication patterns, you can add the SMSESSION token as a credential to the authenticated Sybase Unwired Platform subject for use in single sign-on to SiteMinder-protected systems.

Single Sign-on to a SiteMinder-protected EIS

SiteMinder single sign-on (SSO) provides integration between a SiteMinder-protected EIS and Sybase Unwired Platform.

Table 3. SiteMinder Single Sign-on Integration

Accessed Service	Details
SiteMinder-protected Web service	When an EIS Web service is protected by SiteMinder, Sybase Unwired Platform sends the current Sybase Unwired Platform user's SMSESSION cookie when executing the Web service. For more information on sending the SMSESSION cookie to the SiteMinder-protected Web service, see *Single Sign-on Using NamedCredential* in the *Security* guide.

Accessed Service	Details
SSO2-protected JCo RFC or SAP Web service	The SAP server is configured to use SSO2 tickets for single sign-on. Sybase Unwired Platform sends the user's current MYSAPSSO2 ticket along with the request. SAP validates that the SSO2 ticket is valid and was issued by a trusted peer, and executes the request as that user. **Note:** You must have a SiteMinder agent for SAP installed in a NetWeaver server. Sybase Unwired Platform sends the SMSESSION cookie to NetWeaver, the SAP SiteMinder agent validates the cookie, and then the TokenIssuingLoginModule generates an SSO2 ticket and returns it to Sybase Unwired Platform as a MYSAPSSO2 cookie.
Web service hosted on NetWeaver requiring both SSO2 and SMSESSION	Sybase Unwired Platform sends both SSO credentials when executing the Web service call.

Authentication Cache Timeout and Token Authentication

To reduce the load, Sybase Unwired Platform uses an authentication cache to reduce the load it places on your back-end identity management and security systems. Depending on your security configuration, you can adjust the authentication cache timeout to avoid authentication failures and errors.

By default, the authentication cache holds a user's subject, principals, and credentials used for single sign-on to an EIS for 3600 seconds (one hour). If the user name and password contained inside subsequent Sybase Unwired Platform requests are unchanged, the request is considered authenticated and uses the cached security information for access control and single sign-on to EIS operations.

Note: When using token-based authentication, clients should use a hash code of the token as the password, so Sybase Unwired Platform proceeds through the login modules and replaces the cached token credential. This prevents using an expired token in single sign-on to an EIS.

If you cache an SMSESSION for a user and the token expires before the cache entry, you get authentication failures during the single sign-on EIS operations. This leads to either synchronization errors or operation replay errors.

Configure the authentication cache to avoid errors and failures. If needed, you can disable the authentication cache entirely by setting the cache timeout to 0. Every Sybase Unwired Platform request is reauthenticated. For non-Network Edge basic authentication, you can set the cache interval to slightly less than the Idle Timeout for your SiteMinder session policy.

For Network Edge authentication, you must set the authentication cache timeout to 0. If the URL configured to validate the SMSESSION token also returns an HTTP header with the

expiration time for the token expressed in milliseconds since the epoch (1/1/1970), the HttpAuthenticationLoginModule can use that value to adjust the authentication cache expiration for this subject's entry so it expires at an appropriate time. Use the TokenExpirationTimeHttpHeader to specify the name of the header containing this expiration value. Additionally, you can use TokenExpirationInterval property to reduce time from the expiration so it does not expire while Sybase Unwired Platform is processing a request.

For detailed examples, including how to configure the timeout in the SiteMinder Admin, see *How-To: Set up SUP with SiteMinder* at *http://scn.sap.com/docs/DOC-29574*.

SiteMinder Web Agent Configuration for Sybase® Unwired Platform

When integrating with Sybase Unwired Platform, SiteMinder uses default settings for the Web agent to stop cross-site scripting cross-site scripting (XSS) attacks. The SiteMinder default settings do not allow use of special characters and can lead to integration issues with Sybase Unwired Platform.

By default, the Web agent does not allow certain characters, often seen in XSS attacks, to be including in the URLs it processes. The Web agent allows only legal characters, according to the defined HTTP standard.

Native HTTP OData applications, typically use, and sometimes require, URLs that contain characters within a left and right parenthesis ( ) and within single quotes ' '. The left and right parenthesis and single-quotes characters are prohibited.

The SiteMinder administrator must modify the Web agent configuration in the policy server to either disable XSS filtering entirely or change the default forbidden characters.

Security Configuration to a SiteMinder-protected EIS

With Sybase Unwired Platform, SiteMinder authentication is used in Network Edge and non-Network Edge configurations to authenticate the client of a Web service, SAP JCo, or NetWeaver service.

In your security configuration that integrates with SiteMinder applications, you need a ClientValuePropagatingLoginModule so you can save your SMSESSION cookie as a credential for EIS single sign-on. if the SiteMinder agent adds an sm_user header to client requests, use that header in the ClientValuePropagatingLoginModule to set a user Principal. If the SiteMinder agent does not add an sm_user header, then disable impersonization checking.

You should also have an HttpAuthenticationLoginModule configured for a SiteMinder-protected URL where Sybase Unwired Platform can verify the validity of the user's SMSESSION cookie.

For a detailed example focusing on SiteMinder specific configurations for Sybase Unwired Platform, see *How-To: Set up SUP with SiteMinder* at *http://scn.sap.com/docs/DOC-29574*.

Configuring Security for SiteMinder Token and Basic Authentication

Use Sybase Control Center to create a security configuration for your single sign-on (SSO) applications.

1. In Sybase Control Center, navigate to the **Unwired Platform Cluster** pane and select **Security**.
2. In the **General** tab, click **New** and name your security configuration.
3. Open the **Security** folder and select your configuration. In the **Authentication** tab, click **Add** to add a LoginModule.
4. Choose the **ClientValuePropagatingLoginModule** and add these properties:
   - **Implementation Class** – com.sybase.security.core.ClientValuePropagatingLoginModule
   - **ClientHttpValuesAsPrincipals** – sm_user
   - **ClientHttpValuesAsNamedCredentials** – smsession:SMSESSION2
   - **Control Flag**: optional

   Note: ClientHttpValuesAsNamedCredentials ensures that if the client application picked up an SMSESSION cookie either using Network Edge authentication or an external token, it is saved as a credential named SMSESSION2 on the subject so it can be used for SSO to a SiteMinder-protected EIS. Therefore, the credential.a.name property is SESSION2. Also, ClientHttpValuesAsPrincipals uses the sm_user HTTP header if the client has used Network Edge authentication and enables you to perform impersonation checking.
5. Click **OK**.
6. In the **Authentication** tab, select the default **NoSecLoginModule** and click **Delete**. LoginModule allows logins without credentials, and you must remove it for security integrity.
7. In the **Authentication** tab, click **New** to add a provider.
8. Select and configure the HttpAuthenticationLoginModule:
   a) Select **com.sybase.security.http.HttpAuthenticationLoginModule** and click **Yes** in the Duplicate Authentication Provider warning.
   b) Configure the module's properties so the SiteMinder-protected URL has the same policy server that issued the SMSESSION cookie to the client.
      - ClientValuesToSend = SMSESSION
      - SendClientValuesAs = cookie:SMSESSION

      This causes Sybase Unwired Platform to forward the cookie to the specified SiteMinder-protected URL. If the HTTP status response code is 200, then the SMSESSION cookie is valid and the user is considered authenticated.
9. In the **Authorization** tab, select the **NoSecAuthorizer** provider type and click **Delete**.
10. In the **Settings** tab, adjust the properties as follows:

- **Authentication cache timeout(seconds)** – 0
- **Maximum number of failed authentications** – 5
- **Authentication lock duration(in seconds)** – 600

11. Click **Apply**.
12. In the **General** tab, click **Validate** to check your configuration.
13. With successful validation, click **Apply** to save all changes.

For detailed examples focusing on SiteMinder specific configurations for Sybase Unwired Platform, see *How-To: Set up SUP with SiteMinder* at *http://scn.sap.com/docs/DOC-29574*.

Configuring Network Edge Authentication with a SiteMinder-Protected Web Service

Configure the SMSESSION cookie for Network Edge authentication to a SiteMinder Web service.

Network Edge authentication for SiteMinder requires the Web service endpoint to be changed in Sybase Control Center to use the SMSESSION cookie for single sign-on. By default, the application connection template is configured with the server name set to your SiteMinder-protected server.

1. In the left navigation pane, select **Cluster**, then expand **Domains**, and select the domain for which you want to create a new connection.
2. Select **Connections**.
3. In the right administration pane, select the **Connections** tab.
4. Select the EIS connection pool that is a Web service connection for the SiteMinder-protected service, and click **Properties**.
5. In the **Edit Connection Pool** pane, configure these properties:

Property	Value
credential.a.mapping	Cookie:SMSESSION
credential.a.name	SMSESSION

6. Click **Save**.

For detailed examples focusing on SiteMinder specific configurations for Sybase Unwired Platform, see *How-To: Set up SUP with SiteMinder* at *http://scn.sap.com/docs/DOC-29574*.

Configuring Non-Network Edge Authentication with a SiteMinder-Protected Web Service

Configure the SMSESSION cookie and application connection template for non-Network Edge authentication to a SiteMinder-protected Web service.

Similar to Network Edge authentication for SiteMinder, non-Network Edge authentication requires the Web service endpoint to be changed in Sybase Control Center to use the SMSESSION cookie for single sign-on. However, for non-Network Edge authentication, by

default the application connection template is configured with the server name set to the Sybase Unwired Platform server or a reverse proxy, depending on your configuration.

1. In the left navigation pane, select **Cluster**, then expand **Domains**, and select the domain for which you want to create a new connection.
2. Select **Connections**.
3. In the right administration pane, select the **Connections** tab.
4. Select the EIS connection pool that is a Web service connection pointing to the SiteMinder-protected service, and click **Properties**.
5. In the **Edit Connection Pool** pane, configure these properties:

Property	Value
credential.a.mapping	Cookie:SMSESSION
credential.a.name	SMSESSION

6. Click **Save**.

For detailed examples focusing on SiteMinder specific configurations for Sybase Unwired Platform, see *How-To: Set up SUP with SiteMinder* at *http://scn.sap.com/docs/DOC-29574*.

Single Sign-on Integration Across Client Applications

Administrators can use their single sign-on system (SSO) of choice with Unwired Platform to achieve end-to-end integration across client applications and Enterprise Information Systems (EIS) resources.

In addition to supporting X.509 certificate security, Unwired Platform expands single sign-on support to third-party and standard single sign-on mechanisms. With expanded single sign-on support, Unwired Platform enables the authentication framework to accept HTTP headers and cookies propagated by the client or a proxy server and then authenticate and propagate the user to the EIS.

Network Edge Single Sign-on Authentication

Sybase Unwired Platform applications can integrate with HTTP-based single sign-on (SSO) authentication providers.

Sybase Unwired Platform supports Network Edge authentication by allowing the administrator to configure which client values set in the connection to Sybase Unwired Platform using Network Edge authentication are to be used for authentication into Sybase Unwired Platform server.

Hybrid App and Object API applications can connect to reverse proxy servers or agents at the Network Edge. These agents perform authentication and return authenticated tokens on behalf of those authentication providers to either Unwired Server or HTTP-based enterprise information system (EIS) systems via session personalization values delivered as HTTP cookies, or HTTP headers. An example of an HTTP-based SSO provider is SiteMinder

running inside the enterprise and its SiteMinder agent running at the Network Edge inside an Apache or IIS reverse proxy server.

For more information, see *Single Sign-on* in *Developer Guide: Hybrid Apps*.

Single Sign-on Using NamedCredential

In expanded single sign-on support, Sybase Unwired Platform allows the tokens generated by any system to be used for single sign-on (SSO). Administrators can configure the Web service connection properties with the name of the credential containing the token and how to propagate it to the Web service.

Any login module can add a NamedCredential to the authenticated subject. A NamedCredential is a credential that has a name associated with it and can contain any value. Typically, a credential is used to store a value that can be used to authenticate the user to a backend server using SSO.

The HttpAuthenticationLoginModule by default adds the cookie, when configured to look for one.

To use the NamedCredential added by a login module for single sign-on into EIS, the administrator must set the properties in the EIS connection definition to identify the NamedCredential and how it should be propagated to the EIS in the following format:

```
credential.<X>.name=credential name
```

```
credential.<X>.mapping=credential mapping to header/cookie
```

where X is any unique ID that binds the name and the mapping for a specific credential. Multiple such bindings can be configured so that any or all of the available credentials can be passed to the backend using the specified mechanism.

SiteMinderSSOTokenCredential Example

The following is an example for specifying a sample SiteMinder token from the credential named SiteMinderSSOTokenCredential that should be set in the connection to the backend server as a SMSESSIONID cookie.

```
credential.1.name=SiteMinderSSOTokenCredential
```

```
credential.1.mapping=cookie:SMSESSIONID
```

Propagate Single Sign-on Using ClientValuePropagatingLoginModule

Applications can use session personalization values or HTTP headers and cookies to pass data that should be used for single sign-on into the Enterprise Information System (EIS) backend. The ClientValuePropagatingLoginModule enables administrators to add client values as named credentials, name principals, and role principals to the authenticated subject.

Adding client values as named credentials allows them to be used for single sign-on. When authenticating the user using a token from the client session, if the corresponding login module is unable to retrieve the user name from the token and add it as a principal for use in

impersonation checking, the administrator can configure this provider to add the appropriate header value from the client session as a principal to the authenticated subject.

If there are session personalization values that an application is using as single sign-on data, the values are available to the Web server by using:

- The LoginModule to copy the personalization values or HTTP cookie and header from the client request and attach it to the authenticated subject as a named credential.
- Properties on the connection definition to specify the named credential found on the subject and how to pass it to the Web server.

Note: Rogue applications could intentionally insert HTTP headers with arbitrary values to obtain principals, roles, or credentials that they otherwise would not receive using the other login modules. Use this login module in an environment where you know what the Network Edge behavior and have ensured that applications cannot bypass or override that environment.

To avoid a client setting the client personalization key or HTTP header/cookie value to workaround the impersonation check, only use this configuration when the SSO framework requires it and the deployed applications ensure that the client cannot manipulate the headers set into the session. HTTP headers set by the network edge take precedence over the client personalization key. For more information, see *Impersonation Prevention Using the checkImpersonation Property*.

This login module does not authenticate the subject but adds the NamedCredential if the user is successfully authenticated by other login modules. It always returns “false” from the login method and should always be configured with the controlFlag set to “optional” to avoid affecting the outcome of authentication process. See *controlFlag Attribute Values*.

Table 4. Configuration Options for ClientValuePropagatingLoginModule

Configuration Option	Default Value	Description

ClientHttpValuesAsNamed-Credentials	None	Comma separated list of mappings that specify the names of the client values and the name of the credential to add them. For example:
		`httpHeaderName:credentialName1`
		`httpCookieName:credentialName2`
		`personalizationParameterName1:credentialName3`

Configuration Option	Default Value	Description
ClientHttpValuesAsNamePrincipals	None	Comma separated list of values from the client HTTP map that should be added as name principals after successful authentication.
ClientHttpValuesAsRolePrincipals	None	Comma separated list of values from the client HTTP map that should be added as role principals after successful authentication.

Impersonation Prevention Using the checkImpersonation Property

Administrators can set the **checkImpersonation** property associated with the security configuration to "false" to allow authentication to succeed when in token based authentication the user name presented cannot be matched against any of the user names validated in the login modules.

The **checkImpersonation** property is used when a custom login module that maps the token to a user name and adds a principal with that user name is unavailable. In token-based authentication, even though a valid token may be presented to Unwired Platform, the token may not be associated with the user indicated by the user name. To prevent the user authentication from succeeding, the checkImpersonation property is set to true by default.

When an un-authenticated request is received by Unwired Platform (from a device or DCN request), it may contain a token (in an HTTP header or cookie) that should be validated to authenticate the user. In some cases a user name can be extracted from the token. In Unwired Platform, the specified user name is matched to the name of at least one of the public Principals added by the login modules. If the user name cannot be extracted from the token as part of the validation, then the specified user name is not added as a principal.

In certain situations, it may not be possible for the token validation server to return the user name embedded in the token. If no such custom login module is available, then the administrator can allow authentication to succeed even when the user name presented cannot be matched against any of the user names validated by the configured login modules. In these situations, a custom login module that maps the token to a user name and adds a principal with that user name may be used. To allow this authentication, **set the checkImpersonation property** associated with the security configuration to **false**.

Single Sign-on for SAP

Unwired Platform supports single sign-on (SSO) authentication for mobile clients that access data from an SAP enterprise information system (EIS) using either X.509 certificates or SSO logon tickets (SSO2).

Single sign-on credential support for SAP includes:

- X.509 certificates – use the CertificateAuthenticationLoginModule provider to implement X.509 authentication. At runtime, the mobile client selects the certificate signed by a trusted CA, which is authenticated by the SAP EIS.
- SAP single sign-on (SSO2) tokens – use the HttpAuthenticationLoginModule provider for both basic HTTP authentication and to implement SSO2. At runtime, the client enters a user name/password combination that maps to a user name/password in the SAP EIS. For SSO2, a token is obtained from the configured SAP server using the client-supplied user name/password and is forwarded to other SAP servers configured in the endpoints to authenticate the client, instead of using client-supplied user name/password credentials.

Single Sign-on Authentication

Understand the role of user credentials and X.509 certificates in single sign-on authentication.

Encrypt the communication channel between Unwired Server and the SAP EIS for security reasons:

- For Web services, DOE, and Gateway interactions this requires an HTTPS communication path with mutual certificate authentication. Use Sybase Control Center to navigate to the corresponding connection pool, edit the properties and add the properties "Certificate Alias" (give the name of a certificate alias in the keystore.jks), and "Password" (provide the key password).

During mutual certificate authentication, the client presents a certificate to Unwired Server. In order for authentication to succeed, the client's certificate, or more typically the Certificate Authority (CA) that signed the client certificate must be present in the Unwired Server truststore. Unlike SSO, in a normal JCo connection, the user name is a technical user, and all RFCs are executed in the SAP EIS as that user and not as the end user. The technical user is granted all rights and roles within SAP to allow it to execute the range of RFCs behind the MBOs. However, in the context of SSO to SAP, a technical user certificate is added to the Unwired Server certificate trust store as part of the SNC set up. The technical user certificate is issued by the SAP server and is trusted by the SAP server to impersonate other users. So, once the technical user certificate is authenticated when the SNC connection is established, the SAP server further trusts that the credentials (SSO2 or X.509 values) given to identify the end user are validated by Unwired Server and the SAP server executes the EIS operations as that asserted end-user.

Note: In Sybase Unwired Platform, the password for the CA must match the keystore password (the default `changeit`). When administrators import a certificate to the keystore, they must use the same password for the key alias entry as the keystore password, and thus the same value for the Certificate Alias.

See also

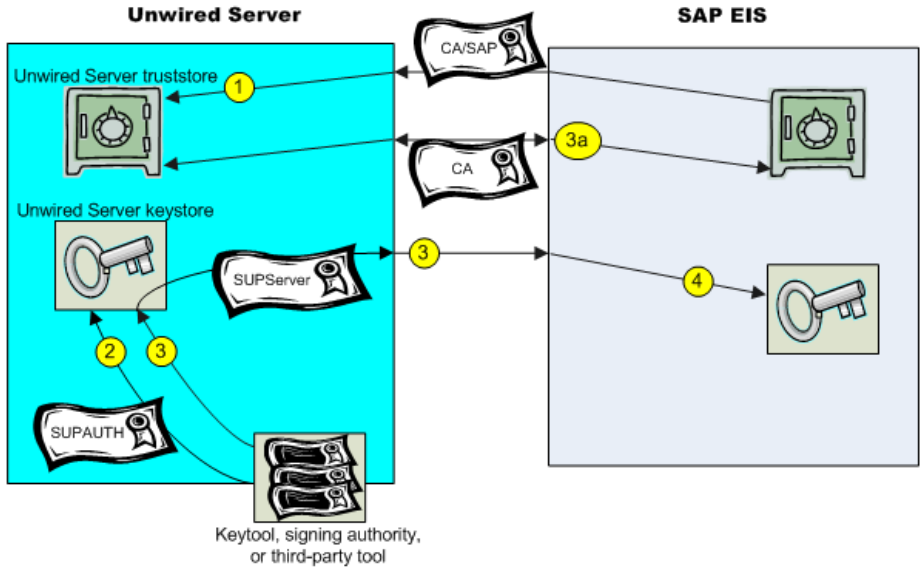
- *Enabling Single Sign-on for DOE-C Packages* on page 22
- *SAP Single Sign-on and DOE-C Package Overview* on page 78
- *SAP Single Sign-on and Online Data Proxy Overview* on page 90
- *Enabling Single Sign-on for OData Applications* on page 23

- *SAP Single Sign-on and Mobile Business Object Package Overview* on page 81
- *Enabling Single Sign-on for Mobile Business Object Packages* on page 24

Configuring X.509 Certificates for SAP Single Sign-on

Import, export, and generate the X.509 certificates that secure communication paths between Unwired Server and the SAP enterprise information system (EIS), and for client authentication, including single sign-on (SSO) with X.509 or SSO2 tokens.

Figure 1: Creating, Importing, and Exporting Certificates



Use Java **keytool** commands to import these certificates into the Unwired Server truststore and keystore. For additional information, see *Using Keytool to Generate Self-Signed Certificates and Keys*.

1. Import SAP CA certificates into the Unwired Server truststore, including:
   - The standard SAP/DOE server root certificate (.crt or .cer) required to establish a trusted relationship between Unwired Server and the SAP EIS.
   - Any CA certificate used to sign .pse certificates used for JCo/SNC communications.
   - For Gateway deployments where Unwired Server is the Online Data Proxy (ODP), import the Gateway server's CA into the truststore of Unwired Platform.

     The ODP requires two certificate files: one that contains the certificate and private key for use by the server, and another that contains only the certificate for use by clients. The certificates should be in the form of a PKCS#10 file using an RSA key pair (key

lengths in the range of 512–16384 are supported), in PEM or DER format. The key usage should be set to Key Encipherment, Data Encipherment, Key Agreement (38).

- Any other required SAP CA certificate. For example, any CA certificate used to sign a client certificate that is to be authenticated by Unwired Server must be imported if you are implementing SSO with X.509.

  Note: If Unwired Server is communicating with a server that is hosting a Web service that is bound to SAP function modules, import that server's CA certificate into the Unwired Server truststore.

For example:

```
keytool -import -keystore SUP_HOME/Servers/UnwiredServer/
Repository/Security/truststore.jks -file <CertificateFile>
```

```
Enter keystore password:  changeit
Trust this certificate? [no]:  yes
```

2. Create a keystore on the Unwired Server host into which you can import the certificate and private key (PKCS #12) issued by the SAP system administrator, then import the certificate into the Unwired Server keystore. This certificate secures communications for packages and is used when a user uses an X.509 certificate rather than a user name and password. For example:

   ```
   keytool -importkeystore -srckeystore SUPAUTH.p12 -
   srcstoretype pkcs12 -srcstorepass <techuserpass> -srcalias
   CERTALIAS -destkeystore SUP_HOME/Servers/UnwiredServer/
   Repository/Security/keystore.jks -deststoretype jks -
   deststorepass changeit -destkeypass changeit
   ```

   Even if the EIS administrator is using the native SAP public-key infrastructure (PKI) to generate certificates, you must still import them into the Unwired Server keystore. The certificate name, *SUPAUTH* and alias, *CERTALIAS* represent the type of package/client to be authenticated, for example:

   - TechnicalUser certificate with doectech alias – a DOE-C package client.
   - SAPUser certificate with SAPClient alias – a SAP or Web service MBO package client.

3. Create and import the SUPServer certificate into the Unwired Server keystore. For example:

   ```
   keytool -importkeystore -srckeystore <source>.p12 -
   srcstoretype pkcs12 -srcstorepass <techuserpass> -srcalias
   <tech user alias> -destkeystore l:/Sybase/UnwiredPlatform/
   Servers/UnwiredServer/Repository/Security/keystore.jks -
   deststoretype jks -deststorepass changeit -destkeypass
   changeit
   ```

Note: (3a)Use the following command to find the srcalias:

```
keytool -list -v -storetype pkcs12 -keystore <source>.p12 -
storepass <source.password>
```

Note: (3b) You can create the SUPServer certificate using Java keytool commands, a third-party tool such as OPENSSL, or the signing authority used to create all SAP server certificates, in which case you need not import any other CA signing authority certificate into the Unwired Server truststore. However, if you create the SUPServer certificate with another CA signing authority, you must import that CA certificate into both the Unwired Server truststore, and into the SAP Server using the STRUST transaction.

4. Import the SUPServer certificate into SAP/DOE server using the STRUST transaction.

You can now configure your environment for mutual authentication and SSO, in which any client connecting to Unwired Server presents credentials, and a server certificate (SUPAUTH) is selected for Unwired Server to present to clients.

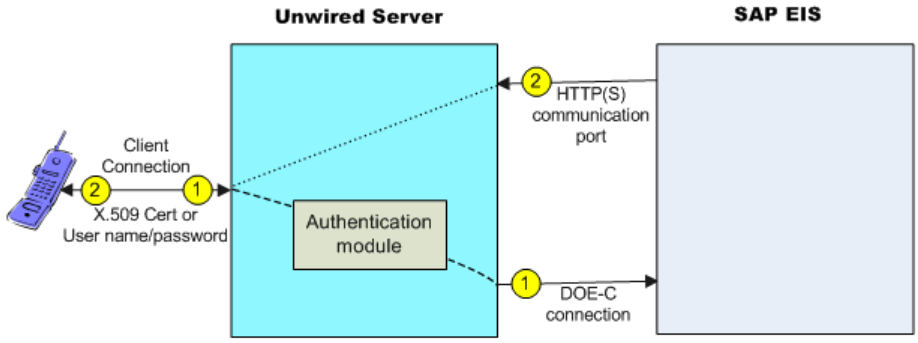
SAP Single Sign-on and DOE-C Package Overview

Understand how DOE-C packages fit in the Unwired Platform landscape, including how to secure communication paths and enable single sign-on (SSO) for these packages.

DOE-C is the connector from Unwired Server to SAP NetWeaver Mobile, which contains the DOE. Sybase Unwired WorkSpace is not used to create MBOs, generate code, create applications, or for deployment. Instead, in DOE-based mobile applications that run in the Unwired Platform environment:

- NetWeaver Mobile handles the data modeling for DOE-C connections.
- Field mappings, connection information, and other application- and package-specific information is defined in the ESDMA, for example the SAP CRM ESDMA, which is deployed to Unwired Server, and automatically converted into an Unwired Platform package by the ESDMA converter.
- DOE-C packages are message-based – NetWeaver Mobile is message-based, and performs queue handling, data caching, and is push-enabled to push data changes out to mobile devices through Unwired Server.

Unwired Server works as a pass-through gateway in the DOE/DOE-C configuration:



1. A DOE-C client application registers with Unwired Server and subscribes to message channels. Unwired Server remembers the push notification information/deviceID/applicationID from the client, but forwards the subscription to DOE through the DOE-C connection (HTTP(S)) to the DOE. When the client performs an operation, that operation flows through Unwired Server via this same connection to the DOE.

   In an SSO configuration, the client provides credentials to Unwired Server (user name and password or X.509 user certificate) that are authenticated by the security configuration's authentication module (CertificateAuthenticationLoginModule for X.509 or HttpAuthenticationLoginModule for SSO2). Once authenticated by Unwired Server, and assuming that Unwired Server and the SAP Server have a secure communication path, SSO is enabled.
2. When application data changes in the SAP EIS and the DOE determines that a particular client has a subscription to that change, DOE connects to the Unwired Server HTTP(S) port and sends a message identifying the client, along with the message payload. Unwired Server looks up the client and queues a message. If the client is connected, the message is delivered immediately. If the client is offline, then Unwired Server attempts to send a push notification to the client (BES HTTP Push for Blackberry, APNS notification for iOS) to attempt to wake up the client and have it retrieve the messages.

   WindowsMobile does not have a separate push notification protocol, so Unwired Server waits for those clients to connect and retrieve their messages.

See also

- *Enabling Single Sign-on for DOE-C Packages* on page 22
- *Single Sign-on Authentication* on page 75

Enabling the DOE-C Connection

Configure the Sybase SAP® Data Orchestration Engine Connector (DOE-C) connection pool between Unwired Server and the SAP EIS. This is the port on which Unwired Server

communicates with the DOE, including forwarding subscriptions and allowing client operations to flow through to the DOE.

This type of connection is available in the list of connection templates only after a DOE-C package has been deployed to Unwired Server.

1. From Sybase Control Center, expand **Domains** > <***DomainName***>, and select **Connections**.

   DomainName is the domain that contains the DOE-C package.
2. Select an existing connection pool, and set the property values required to enable an authenticated HTTPS connection to the DOE.

   If defining a security profile to implement mutual authentication with basic authentication, add the `certificateAlias` property, which overrides the technical user name and password fields. The technical user name and password fields can be empty, but only if `certificateAlias` is set. The specified certificate is extracted from the Unwired Server keystore and supplied to the DOE.
3. Select **Save**.

Deploying and Configuring DOE-C Packages

Unlike Hybrid App or MBO packages that use Sybase Control Center to deploy packages to Unwired Server, you must deploy the DOE-C package to specific domain using the DOE-C command line utility (CLU). Once deployed, the DOE-C package is visible and manageable from Sybase Control Center.

1. Start the command line utility console. See *Starting the Command Line Utility Console* in *System Administration*.
2. Deploy the DOE-C package. During deployment, you can set the domain and security configuration using the **setPackageSecurityConfiguration** command with -d and -sc options. After deployment, you can set the security configuration using the **setPackageSecurityConfiguration** command, or perform this task later from Sybase Control Center.

   See *SAP DOE Connector Command Line Utility* in the *System Administration* guide.

Next

Verify or set the security configuration for the domain or package.

See also

- *Security Configurations That Implement Single Sign-on Authentication* on page 92

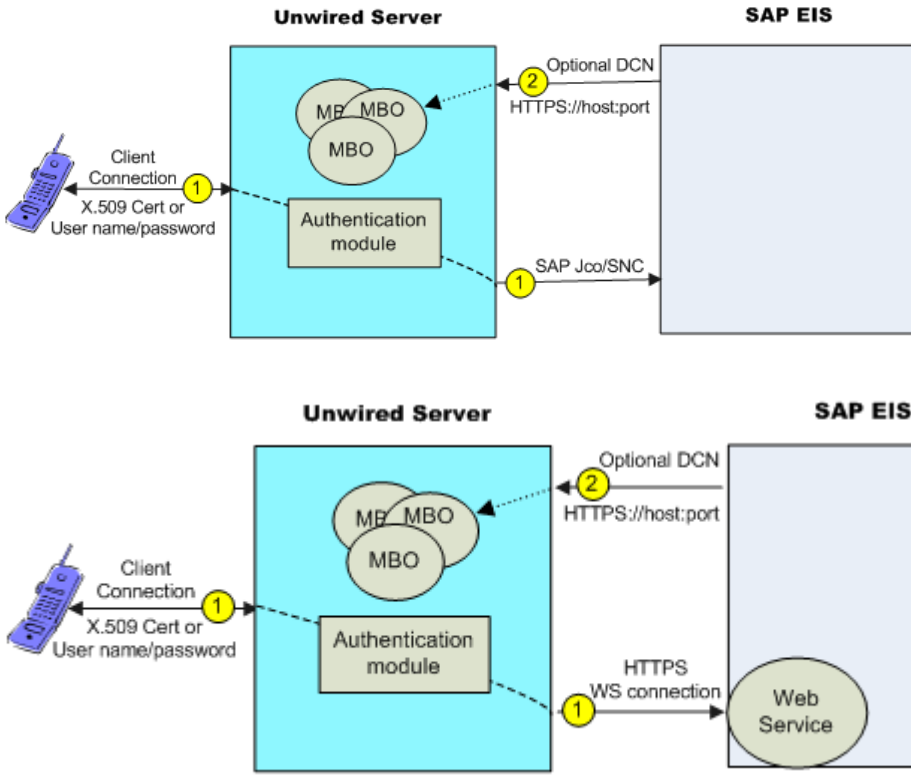
SAP Single Sign-on and Mobile Business Object Package Overview

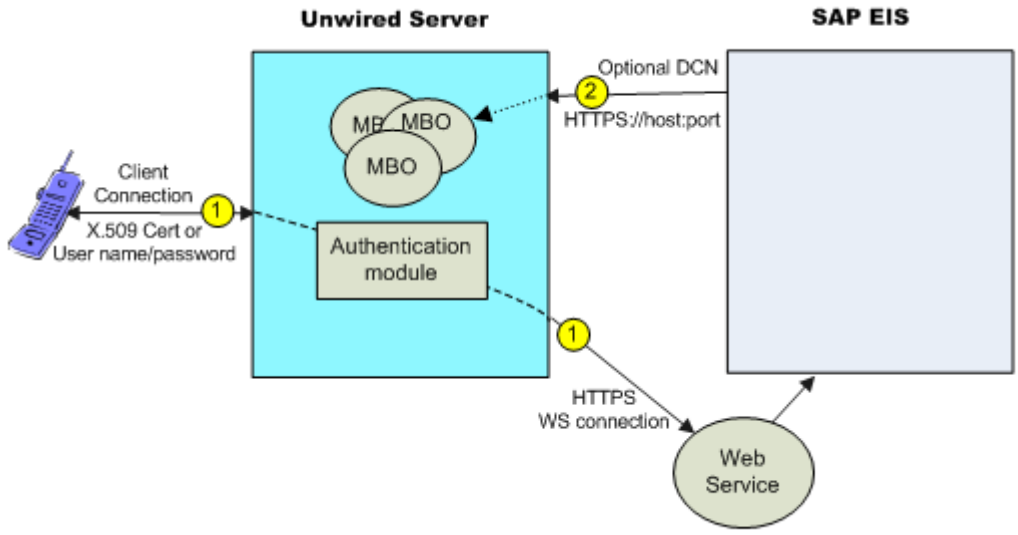
Understand how to secure communication ports and enable single sign-on (SSO) for packages that contain mobile business objects (MBOs) bound to an SAP enterprise information system (EIS).

SAP MBOs bound directly to SAP BAPIs and RFCs, as well as SAP BAPIs exposed as Web services. Once deployed, Unwired Platform supports Java connector (JCo) connections and Secure Network Communications (SNC) for SAP MBOs, and HTTP(S) connections to Web services.

Once deployed, connection information, and other application- and package-specific information is maintained by Unwired Server. Unwired Server packages that contain SAP MBOs support message-based and replication-based applications and perform queue handling, data caching, and synchronization services.

Typical data flow for SAP MBO packages that use data change notification (DCN) as a refresh mechanism:





1. Data flows from Unwired Server to the EIS through a configured connection pool. For secure connections:
   - Jco – communicates with the SAP EIS using the SAP JCo proprietary communication protocol. Optionally use SNC if required for your installation.
   - Web service – communicates to the Web service host using HTTPS, whether the Web service is on the same server that hosts the SAP BAPIS/RFCs to which the Web service is bound, or a different server.

     In an SSO configuration, the client provides credentials to Unwired Server (username and password or X.509 user certificate) that are authenticated by the security configuration's authentication module (CertificateAuthenticationLoginModule for X.509 or HttpAuthenticationLoginModule for SSO2). Once authenticated by Unwired Server, and assuming that Unwired Server and the EIS have a secure communication path, SSO is enabled.
2. (Optional) Configure a data change notification (DCN) port if this is the data refresh policy for any of the MBOs within the package.

See also

- *Enabling Single Sign-on for Mobile Business Object Packages* on page 24
- *Single Sign-on Authentication* on page 75

Single Sign-on for SAP MBO Package Prerequisites

Before implementing SSO for SAP MBO packages, configure the MBOs so client credentials can be propagated to the EIS and, if enabling SSO for a Hybrid App application, add the appropriate starting point.

Configure the MBO. See these topics in *Sybase Unwired WorkSpace - Mobile Business Object Development*:

- *Propagating a Client's Credentials to the Back-end Data Source*
- *Modify SAP Connection Properties* – for SAP MBOs
- *Configuring an SAP Exposed Web Service MBO to Use Credentials* – for SAP function modules exposed as Web services

Configure the Hybrid App to use SSO2 or X.509 credentials by adding and configuring a credential starting point for the Hybrid App. See *Configuring the Workflow Application to Use Credentials* in the *Developer Guide: Hybrid Apps* for details.

Single Sign-on for SAP MBO Package Postrequisites

After configuring SSO for SAP MBO packages on Unwired Server, install certificates on the mobile device and test them.

- Hybrid App applications – see *Installing and Testing X.509 Certificates on Simulators and Mobile Devices* in the *Developer Guide: Hybrid Apps*.
- Native applications – install and import X.509 certificates and use the Object API to select them for client connections. Refer to your platform's *Developer Guide* for details:
  - *Installing and Testing X.509 Certificates on Simulators and Mobile Devices*
  - *Single Sign-On With X.509 Certificate Related Object API*

Creating Connections and Connection Templates

Create a new connection or connection template that defines the properties needed to connect to a new data source.

1. In the left navigation pane, expand the **Domains** folder, and select the domain for which you want to create a new connection.
2. Select **Connections**.
3. In the right administration pane:
   - To create a new connection – select the **Connections** tab, and click **New**.
   - To create a new connection template – select the **Templates** tab, and click **New**.
4. Enter a unique **Connection pool name** or template name.
5. Select the **Connection pool type** or template type:
   - JDBC – choose this for most database connections.

- Proxy - choose this if you are connecting to a proxy endpoint; for example, an Online Data Proxy data source or other proxy endpoint.
- WEBSERVICE – choose this if you are connecting to a Web Services (SOAP or REST) data source.
- SAP – choose this if you are connecting to an SAP (JCO) data source.

6. Select the appropriate template for the data source target from the **Use template** menu. By default, several templates are installed with Unwired Platform; however, a production version of Unwired Server may have a different default template list.
7. Template default properties appear, along with any predefined values. You can customize the template, if required, by performing one of:

   - Editing existing property values – click the corresponding cell and change the value that appears.
   - Adding new properties – click the **<ADD NEW PROPERTY>** cell in the Property column and select the required property name. You can then set values for any new properties.

   Note: In a remote server environment, if you edit the sampledb Server Name property, you must specify the remote IP number or server name. Using the value "localhost" causes cluster synchronization to fail.
8. Test the values you have configured by clicking **Test Connection**. If the test fails, either values you have configured are incorrect, or the data source target is unavailable. Evaluate both possibilities and try again. Only the SAP and JDBC connection pool types can test the connection. The Proxy and WEBSERVICE pool types cannot test the connection.
9. Click **OK** to register the connection pool.
   The name appears in the available connection pools table on the Connections tab. Administrators can now use the connection pool to deploy packages.

Configuring an SAP Java Connector With SNC

Create a Java Connection (JCo) to an SAP Server in Unwired Server from Sybase Control Center where SNC is required.

Prerequisites

Start Unwired Server services and log in to Sybase Control Center as the administrator, and download and install the SAP cryptographic libraries.

Task

The SAP JCo connection provides access for various client types, including those that use SSO2 tokens and X.509 certificates.

1. Expand the cluster, expand the **Domains** folder, expand the domain to which the package is to be deployed, and select **Connections**.

2. Select the **Connections** tab and click **New**. Name the connection pool `SAP Server`, select **SAP** as the Connection pool type, select the **SAP template**, and enter appropriate properties for the SAP enterprise information system (EIS) to which you are connecting. For example:

   Alternatively, if you require a template with these SNC properties prepopulated for future convenience, create a new template for SNC-enabled SAP connections, and then use that template for these properties.

   - Language (jco.client.lang) = `EN`
   - Logon User (jco.client.user)=`snctest`
   - Password (jco.client.password) =`********` (snctest user password)
   - Host name (jco.client.ashost) = `sap-doe-vm1.sybase.com`
   - System number (jco.client.sysnr) = `00`
   - SNC mode (jco.client.snc_mode) = `1`
   - SNC name (jco.snc_myname) = `p:CN=SNCTEST, O=Sybase, L=Dublin, SP=California, C=US`
   - SNC service library path (jco.client.snc_lib) = `C:/sapcryptolib/sapcrypto.dll` (the location of the cryptographic library)
   - Client number (jco.client.client) = `100`
   - SNC partner (jco.client.snc_partnername) = `p:CN=sap-doe-vm1, OU=SUP, O=Sybase, C=US`
   - SNC level (jco.client.snc_qop) = `1`

3. Click **Test Connection** to verify access to the SAP server, and click **OK**.

Generating and Installing a PSE Certificate on Unwired Server

Generate a PSE certificate on Unwired Server to use in testing connections with SAP Systems when using the SAP Cryptographic Library to secure the connection using Secure Network Communications (SNC).

Prerequisites

Download and install the SAP Cryptographic Library.

Task

These instructions describe how to generate an X.509 certificate for testing SAP JCo and single sign-on with SNC only. In a production environment, a different entity controls certificate management. For example, an SAP system administrator controls certificate generation and management for his or her particular environment, including maintaining the certificate list in a Personal Security Environment (PSE) with trust manager.

Note: When the CertificateAuthenticationLoginModule gets a certificate from a client, it can optionally validate that it is a trusted certificate. The easiest way to support validation is to import the CA certificate into the *`SUP_HOME`*`\Servers\UnwiredServer`

`\Repository\Security\truststore.jks` file, which is the default Unwired Server truststore.

Use the SAPGENPSE utility to create a PSE certificate to use for testing. See *http://help.sap.com/saphelp_nw04s/helpdata/en/a6/f19a3dc0d82453e10000000a114084/content.htm*. The basic steps are:

1. Generate the certificate from the SAP Cryptograhpic Library directory. For example,`C:\sapcryptolib`:

   ```
   sapgenpse get_pse <additional_options> -p <PSE_Name> -r
   <cert_req_file_name> -x <PIN> <Distinguished_Name>
   ```

2. Copy the PSE certificate (for example, `SNCTEST.pse`) to the location of your installed SAP Cryptographic Library. For example, `C:\sapcryptolib`.
3. Generate a credential file (cred_v2) from the `C:\sapcryptolib` directory:

   ```
   sapgenpse seclogin -p SNCTEST.pse -O DOMAIN\user -x
   password
   ```

   Note: The user that generates the credential file must have the same user name as the process (that is, either `mlserv32.dll` or `eclipse.exe`) under which the Unwired Platform service runs. The user must also be user of the domain as determined with the `-O` *`DOMAIN\user`* flag.

SAP Java Connector Properties

Configure SAP Java Connector (JCo) connection properties.

For a comprehensive list of SAP JCo properties you can use to create an instance of a client connection to a remote SAP system, see *http://help.sap.com/javadocs/NW04/current/jc/com/sap/mw/jco/JCO.html#createClient(java.util.Properties)*.

This list of properties can be used by all datasource types. Sybase does not document all native endpoint properties. However, you can add native endpoint properties, naming them using this syntax:

```
<NativeConnPropName>=<SupportedValue>
```

Table 5. General connection parameters

Name	Description	Supported values
Enable ABAP Debugging	Enables or disables ABAP debugging. If enabled, the connection is opened in debug mode and the invoked function module can be stepped through in the debugger. For debugging, an SAP graphical user interface (SAPGUI) must be installed on the same machine the client program is running on. This can be either a normal Windows SAPGUI or a Java GUI on Linux/UNIX systems.	Not supported. Do not set this parameter or leave it set to 0.
Remote GUI	Specifies whether a remote SAP graphical user interface (SAPGUI) should be attached to the connection. Some older BAPIs need an SAPGUI because they try to send screen output to the client while executing.	Not supported. Do not set this parameter or leave it set to 0.
Get SSO Ticket	Generates an SSO2 ticket for the user after login to allow single sign-on. If RfcOpenConnection() succeeds, you can retrieve the ticket with RfcGetPartnerSSOTicket() and use it for additional logins to systems supporting the same user base.	Not accessible by the customer. Do not set this parameter or leave it set to 0.
Use X509	Unwired Platform sets this property when a client uses an X509 certificate as the login credential.	If an EIS RFC operation is flagged for SSO (user name and password personalization keys selected in the authentication parameters) then Sybase Unwired Platform automatically sets the appropriate properties to use X.509, SSO2, or user name and password SSO credentials. The corresponding properties should not be set by the administrator on the SAP endpoint.

Name	Description	Supported values
Additional GUI Data	Provides additional data for graphical user interface (GUI) to specify the SAProuter connection data for the SAPGUI when it is used with RFC.	Not supported.
GUI Redirect Host	Identifies which host to redirect the remote graphical user interface to.	Not supported.
GUI Redirect Service	Identifies which service to redirect the remote graphical user interface to.	Not supported.
Remote GUI Start Program	Indicates the program ID of the server that starts the remote graphical user interface.	Not supported.

Web Services Properties

Configure connection properties for the Simple Object Access Protocol (SOAP) and Representational State Transfer (REST) architectures.

Name	Description	Supported Values
Password	Specifies the password for HTTP basic authentication, if applicable.	Password
Address	Specifies a different URL than the port address indicated in the WSDL document at design time.	HTTP URL address of the Web service
User	Specifies the user name for HTTP basic authentication, if applicable.	User name
Certificate Alias	Sets the alias for the Unwired Platform keystore entry that contains the X.509 certificate for Unwired Server's SSL peer identity. If you do not set a value, mutual authentication for SSL is not used when connecting to the Web service.	Use the alias of a certificate stored in the Unwired Server certificate keystore.

Name	Description	Supported Values
authentication-Preemptive	When credentials are available and this property is set to the default of false, this property allows Unwired Server to send the authentication credentials only in response to the receipt of a server message in which the HTTP status is 401 (UNAUTHORIZED) and the WWW-Authenticate header is set. In this case, the message exchange pattern is: request, UNAUTHORIZED response, request with credentials, service response. When set to true and basic credentials are available, this property allows Unwired Server to send the authentication credentials in the original SOAP or REST HTTP request message. The message exchange pattern is: request with credentials, a service response.	False (default) True
Socket Timeout	The socket timeout value controls the maximum time in milliseconds after a web service operation (REST or SOAP) is allowed to wait for a response from the remote system; if the EIS system does not respond in that time, the operation fails and the SUP thread is unblocked.	Time in milliseconds (default: 6000). Range of [0 – 2147483647], where 0 is interpreted as infinity.
credential.<X>.name	Defines the EIS connection definition to identify the NamedCredential **Note:** For more information on token-based SSO using NamedCredential, see *Single Sign-on Using NamedCredential*.	• `credential.<X>.name=credential name` • `header:<HTTP header name>` • `cookie: <HTTP cookie name>`

Name	Description	Supported Values
credential.<X>.mapping	Defines how the NamedCredential should be propagated to the EIS. **Note:** For more information on token-based SSO using NamedCredential, see *Single Sign-on Using NamedCredential*.	`credential.<X>.mapping=credential mapping to header/cookie`

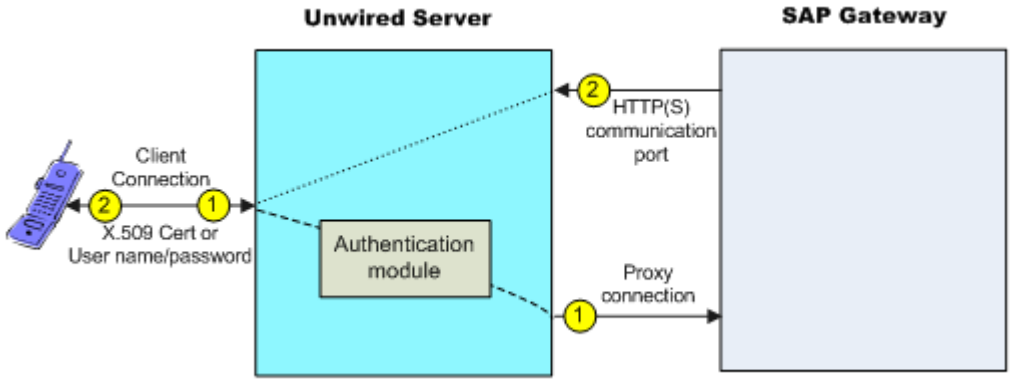
SAP Single Sign-on and Online Data Proxy Overview

Understand how OData applications fit in the Unwired Platform landscape and learn how to secure communication paths and enable single sign-on (SSO) for these applications.

The proxy connector is the online data proxy (ODP) connector between OData applications and the SAP Gateway, and uses an HTTP(S) connection from Unwired Server to the SAP Gateway. A separate HTTP(S) port is used by the SAP Gateway to push changes through Unwired Server to the OData application. Sybase Unwired WorkSpace is not used to create MBOs, generate code, create applications, or for deployment. Instead, in OData-based mobile applications that run in Unwired Server:

- Applications are developed using the OData SDK.
- The SAP Gateway/enterprise information system (EIS) is responsible for data federation and content management.
- OData applications are message based – the SAP Gateway performs queue handling, data caching, and is push-enabled to push data changes out to Unwired Server, which in turn pushes these changes to the physical devices.

Unwired Server acts as a pass-through server for OData-based applications:



1. An OData client application registers with Unwired Server and subscribes to push notifications from the SAP Gateway. Unwired Server forwards the subscription request to

the SAP Gateway. The SAP Gateway stores the subscription request for the collection with the push delivery address (HTTP(S) SSL Port).

In an SSO configuration, the client provides credentials to Unwired Server (user name and password or X.509 user certificate) that are authenticated by the security configuration's authentication module (CertificateAuthenticationLoginModule for X.509 or HttpAuthenticationLoginModule for SSO2). Once authenticated by Unwired Server, and assuming that Unwired Server and the SAP Gateway have a secure communication path, SSO is enabled.

2. When application data changes in SAP and determines that a particular client has a subscription to that change, the Gateway connects to the Unwired Server HTTP(S) port and sends a message identifying the client, along with the message payload. Unwired Server looks up the client and queues the message. If the client is connected, the message is delivered immediately. If the client is offline, then Unwired Server attempts to send a push notification to the client (BES HTTP Push for Blackberry, APNS notification for iOS) to attempt to wake up the client and have it retrieve the messages.

See also

- *Single Sign-on Authentication* on page 75
- *Enabling Single Sign-on for OData Applications* on page 23

Preparing the SAP Gateway

Configure the SAP Gateway to push OData application data to Unwired Server, including configuring the RFC destination for HTTPS on the Gateway.

1. Log on to a Gateway system and go to transaction **sm59**.
2. Create a RFC connection of type **G**.
3. Enter the domain name of Unwired Server (CN of the Unwired Server certificate) in the **Target Host** field .
4. Enter `/GWC/SUPNotification` in the **Path Prefix** field.
5. Enter 8004 in the **Service No.** field.
6. Select the **Logon & Security** tab.
7. Under **Security Options**, click the SSL **Active** option.
8. Select **Default SSL Client (Standard)** from the **SSL Certificate** drop-down list.
9. Click **Save**, then **Connection Test**.

   The test should be successful, a HTTP OK success message displays.

Note: Change the push endpoint in the proxy property of the application templates to https://<domain name of the Unwired Server>:<SSL Port>/GWC/SUPNotification/. In this example, 8004 is the SSL port to which the Gateway pushes data changes:

```
https://inln50089324a.dhcp.blrl.sap.corp:8004/GWC/
SUPNotification/
```

Preparing Your SAP Environment for Single Sign-on

Verify that the SAP® enterprise information system (EIS) is configured correctly to accept SSO connections from Unwired Server.

1. Set all parameters for the type of credentials accepted by the server:
   - SSO2 token – verify everything is set properly with the SSO2 transaction.
   - X.509 certificate – set up, import, and verify certificates using the Trust Manager (transaction STRUST).
2. Use the ICM configuration utility to enable the ICM HTTPS port.
3. Set the type of authentication to enable communication over HTTPS.
   - Server authentication only – the server expects the client to authenticate itself using basic authentication, not SSL
   - Client authentication only – the server requires the client to send authentication information using SSL certificates. The ABAP stack supports both options. Configure the server to use SSL with client authentication by setting the ICM/HTTPS/verify_client parameter:
     - 0 – do not use certificates.
     - 1 – allow certificates (default).
     - 2 – require certificates.
4. Use the Trust Manager (transaction STRUST) for each PSE (SSL server PSE and SSL client PSE) to make the server's digitally signed public key certificates available. Use a public key infrastructure (PKI) to get the certificates signed and into the SAP system.

   There are no SSO access restrictions for MBO data that span multiple SAP servers.

   See SAP product documentation at *http://help.sap.com/saphelp_aii710/helpdata/en/49/23501ebf5a1902e10000000a42189c/frameset.htm* for information about the SAP Trust Manager.
5. To enable secure communication, Unwired Server and the SAP server that it communicates with must exchange valid CA X.509 certificates. Deploy these certificates, which are used during the SSL handshake with the SAP server into the Unwired Server truststore.
6. The user identification (distinguished name), specified in the certificate must map to a valid user ID in the AS ABAP, which is maintained by the transaction SM30 using table view (VUSREXTID).

See *Configuring the AS ABAP for Supporting SSL* at *http://help.sap.com/saphelp_aii710/helpdata/en/49/23501ebf5a1902e10000000a42189c/frameset.htm*

Security Configurations That Implement Single Sign-on Authentication

Use the CertificateAuthenticationLoginModule authentication module to implement X.509 authentication or HttpAuthenticationLoginModule to implement SSO2.

Creating and Assigning a Security Configuration That Uses SSO2 Tokens

Create a new security configuration, assign the HttpAuthenticationLoginModule authentication provider to it, and assign the security configuration to an Unwired Server domain or package.

The HttpAuthenticationLoginModule authentication provider supports SSO2 token logins to SAP systems through JCo and Web service connections, DOE-C packages, and other packages that require token authentication.

1. Create the new security configuration:
   a) From Sybase Control Center, select **Security**.
   b) Select the **General** tab, click **New**, and enter a name for the new security configuration, for example, `SAPSSOSECADMIN`. Click **OK**.
2. Configure the SAP EIS portal:
   a) Apply SAP Note 1250795 to the portal server. This is required to get the HTTP challenge pop-up window.
   b) Verify the SAP EIS URL configured as the Unwired Server SAP Server URL property is an URL with a challenge popup window, not just a generic portal URL.
   c) Maintain the URL and control flag security configuration parameters, which are the only required parameters.
3. Configure the new security configuration:
   a) Select the **SAPSSOSECADMIN** security configuration.
   b) Select the **Authentication** tab.
   c) Click **New** and select **com.sybase.security.http.HttpAuthenticationLoginModule** as the authentication provider. Set the SAP server URL, the SSO cookie name (typically set to `MYSAPSSO2`), and other properties as appropriate for the connection.
4. Select the **General** tab, and click **Validate** to confirm that Unwired Server accepts the new security configuration.
   A message indicating the success of the validation appears above the menu bar.
5. Click **Apply** to save changes to the security configuration, and apply them across Unwired Server.
6. Assign the SAPSSOSECADMIN security configuration to the domain to which SSO packages are being deployed.
   a) Click **Domains** > ***DomainName*** > **Security** .
   b) Click **Assign**.
   c) Select **SAPSSOSECADMIN** and click **OK**.
7. If any other security configurations have been assigned to this SSO domain, Sybase suggests that you unassign them.

   However, many deployments of Unwired Platform do mix SSO and non-SSO MBOs or operations in the same package. There are certain operations that are not sensitive and do not require the overhead of setting up the SSO connection to the backend. Some packages

may even perform DCNs, and the DCN user would not be part of the SSO-enabled login module. If you do authenticate a user against a non-SSO login module and then attempt to perform an SSO-enabled operation, then the credentials are sent to the backend, which may not be desired.

Creating and Assigning a Security Configuration That Uses X.509 Credentials

Create a new security configuration, assign the CertificateAuthenticationLoginModule authentication provider to it, and assign the security configuration to an Unwired Server domain or package.

The CertificateAuthenticationLoginModule authentication provider supports X.509 certificate logins to SAP systems through JCo, DOE-C, Online Data Proxy, and Web service connections. You can assign security configurations to domains, packages, or applications.

1. Create the new security configuration:
   a) From Sybase Control Center, select **Security**.
   b) Select the **General** tab, click **New**, and enter a name for the new security configuration, for example, `X509SECADMINCERT`. Click **OK**.
2. Configure the new security configuration:
   a) Expand the Security folder.
   b) Select the **X509SECADMINCERT** security configuration.
   c) Select **Authentication**.
   d) Select **New**.
   e) Select **com.sybase.security.core.CertificateAuthenticationLoginModule** as the Authentication provider.
   f) Click **OK** to accept the default settings, or modify any of these settings as required:
      - Click **<Add New Property>**, select **Validate Certificate Path** and set the value to **true**.
      - If more than one truststore is defined in Unwired Server, click **<Add New Property>**, select **Trusted Certificate Store** and set the value to the location of the Java truststore that contains the Unwired Server trusted CA certificates. Otherwise, the default Unwired Server truststore is used.
      - If you change the default password for the truststore, click **<Add New Property>**, select **Trusted Certificate Store Password** and set the value of the truststore password.
   g) Click **OK**.
3. Select the **General** tab, select **Validate**, then **Apply**.
4. Assign the X509SECADMINCERT security configuration to an Unwired Server domain. This example uses the default domain, but you can specify any domain to which the package is deployed:
   a) Click **Domains** > ***DomainName*** > **Security** .

b) Click **Assign**.
c) Select **X509SECADMINCERT** and click **OK**.

5. If any other security configurations have been assigned to this SSO domain, Sybase suggests that you unassign them.

   However, many deployments of Unwired Platform do mix SSO and non-SSO MBOs or operations in the same package. There are certain operations that are not sensitive and do not require the overhead of setting up the SSO connection to the backend. Some packages may even perform DCNs, and the DCN user would not be part of the SSO-enabled login module. If you do authenticate a user against a non-SSO login module and then attempt to perform an SSO-enabled operation, then the credentials are sent to the backend, which may not be desired.

Creating Security Profiles to Enable Mutual Authentication for SAP

Create security profiles and associate them with X.509 server certificates that can be used to establish secure connections between a client, Unwired Server, and the SAP EIS.

Prerequisites

- Your SAP system must be configured for HTTPS mutual authentication
- Import the third party's private-key certificate used by Unwired Server to mutually authenticate the client into the Unwired Server keystore:
  - *SUPServer* certificate – represents the certificate used to secure an HTTPS connection between Unwired Server and SAP Server or other enterprise information system (EIS), where data and information flow from Unwired Server to the EIS, which could be a DOE-C, Web Service, or Proxy connection.
  - *SAPServer* certificate – represents the certificate used to secure the communication path between the SAP Server or EIS and Unwired Server, where data and information flow from the EIS to Unwired Server on an HTTPS port (8001, 8002, and so on), which are made available to the EIS for pushing data to Unwired Server. For SAP Servers, this could be NetWeaver/DOE (TechnicalUser), or the SAP Gateway.

Task

To secure connections, create two new security profiles: one for the SAP gateway and one for Unwired Server. If you imported the user and CA certificates into keystore or truststore locations other than the default, make sure the paths and passwords reflect them.

1. In the Sybase Control Center navigation pane, click **Configuration**.
2. From the **General** tab, click **SSL Configuration**.
3. Select **<ADD NEW SECURITY PROFILE>** and create a security profile for SAP servers:
   - Security profile name – for example, `TechnicalUser` for NetWeaver/DOE connections or `Proxy` for SAP Gateway connections.

- Certificate alias – the case sensitive certificate alias you defined when you imported the certificate into the keystore. For example, `doetech`, `proxy` (or whatever value you set the alias to using the **keytool -alias** option).
- Authentication – `strong_mutual`

4. Select **<ADD NEW SECURITY PROFILE>** and create an Unwired Server security profile:
   - Security profile name – for example, `SUPServer`.
   - Certificate alias – `SUP` (or whatever value you set the alias to using the **keytool -alias** option).
   - Authentication – `strong_mutual`.
5. Restart Unwired Server.

Enabling the HTTPS Port and Assigning the Unwired Server Security Profile

Enable an HTTPS port for secure communication between Unwired Server and the SAP EIS.

For DOE-C and SAP Gateway, this is the port to which the DOE or Gateway connects to Unwired Server and sends a message identifying the client, along with the message payload.

1. In the Sybase Control Center navigation pane, select **Configuration**.
2. In the right administration pane, click the **Web Container** tab, click **Communication Ports**.
3. Expand **Show secure data change notification ports**.
4. Select a port number and enter:
   - Status – `enabled`.
   - SSL Security Profile – `SUPServer` or whatever you named the security profile associated with the Unwired Server user certificate.

   Note: You can add a new HTTPS port if you do not want to use 8001 or 8002. For Gateway connections, the port must match that of the port you defined in the Gateway, for example 8004. See *Preparing the SAP Gateway*.
5. Select **Save**.
6. Restart Unwired Server.

See also

- *Preparing Your SAP Environment for Single Sign-on* on page 92

Distributing Single Sign-on Related Files in an Unwired Server Cluster

Place required files in the appropriate primary Unwired Server subdirectory so they are distributed to all Unwired Servers within the cluster during cluster synchronization.

Any changes to a named security configuration affect the cluster and trigger a cluster synchronization, which automatically zips the files in the primary Unwired Server `CSI`

subdirectory and distributes them to the other servers in the cluster. Copy all certificate and other security-related files to the `CSI` subdirectory.

The provider configuration information, which includes the server certificate file name and location, must be the same on all cluster nodes. The same is true for the cryptographic DLLs and certificate files for SSO using X509.

1. On the primary server in the cluster, put any SAP certificate files or truststores into the *SUP_HOME*`\Servers\UnwiredServer\Repository\CSI\conf` directory.

   Use system properties to specify the full path and location of the file in the configuration so they can be accessed from different servers within the cluster if installation directories are different from that of the primary server. For example:

   ```
   ${djc.home}/Repository/CSI/conf/
   SNCTEST.pse
   ```

   For X.509 CertificateAuthenticationLoginModule, if the *ValidateCertificatePath* is set to true, the default, the CA certificate (or one of its parents) must be installed in the truststore for each server.

   Note: Unwired Server truststore and keystore files:
   - *SUP_HOME*`\Servers\UnwiredServer\Repository\Security\truststore.jks` – is the Unwired Server trust store that contains CA (or parent) certificates. Unwired Server trusts all CA or parent certificates in `truststore.jks`.
   - *SUP_HOME*`\Servers\UnwiredServer\Repository\Security\keystore.jks` – contains client certificates only.

   The CertificateAuthenticationLoginModule also has `Trusted Certificate Store*` and `Store Password` properties which you can to keep the module out of the default Unwired Server trust store. You must first:

   a) Use **keytool** to put the CA certificate into a new keystore.
   b) Put the keystore into the `Repository\CSI\conf` subdirectory.
   c) Include the path in the `Trusted Certificate Store` property.
2. From Sybase Control Center, add the login module.
3. Restart all Unwired Server within the cluster.

Stacking Providers and Combining Authentication Results

Optionally, implement multiple login modules to provide a security solution that meets complex security requirements. Sybase recommends provider stacking as a means of eliciting more precise results, especially for production environment that require different authentications schemes for administrators, DCN, SSO, and so on.

Stacking is implemented with a controlFlag attribute that controls overall behavior when you enable multiple providers. Set the controlFlag on a specific provider to refine how results are processed.

For example, say your administrative users (supAdmin in a default installation) are not also users in an EIS system like SAP. However, if they are authenticated with just the default security configuration, they cannot also authenticate to the HttpAuthenticationLoginModule used for SSO2Token retrieval. In this case, you would stack a second login modules with a controlFlag=sufficient login module for your administrative users.

Or, in a custom security configuration (recommended), you may also find that you are using a technical user for DCN who is also not an SAP user. This technical user does not need SSO because they will not need to access data. However, the technical user still needs to be authenticated by Unwired Server. In this case, you can also stack another login module so this DCN user can login.

1. Use Sybase Control Center to create a security configuration and add multiple providers as required for authentication.
2. Order multiple providers by selecting a login module and using the up or down arrows at to place the provider correctly in the list.

   The order of the list determines the order in which authentication results are evaluated.
3. For each provider:
   a) Select the provider name.
   b) Click **Properties**.
   c) Configure the controlFlag property with one of the available values: required, requisite, sufficient, optional.

      See *controlFlag Attribute Values* for descriptions of each available value.
   d) Configure any other common security properties as required.
4. Click **Save**.
5. Select the **General** tab, and click **Apply**.

For example, say you have sorted these login modules in this order and used these controlFlag values:

- LDAP (required)
- NT Login (sufficient)
- SSO Token (requisite)
- Certificate (optional)

The results are processed as indicated in this table:

Provider	Authentication Status							
LDAP	pass	pass	pass	pass	fail	fail	fail	fail
NT Login	pass	fail	fail	fail	pass	fail	fail	fail

Provider	Authentication Status							
SSO Token	*	pass	pass	fail	*	pass	pass	fail
Certificate	*	pass	fail	*	*	pass	fail	*
Overall result	pass	pass	pass	fail	fail	fail	fail	fail

See also

- *LDAP Security Provider* on page 56
- *LDAP Configuration Properties* on page 153

controlFlag Attribute Values

(Not applicable to Online Data Proxy) The Sybase implementation uses the same control flag (controlFlag) attribute values and definitions as those defined in the JAAS specification.

If you stack multiple providers, you must set the control flag attribute for each enabled provider.

Control Flag Value	Description
Required	The LoginModule is required. Authentication proceeds down the LoginModule list.
Requisite	The LoginModule is required. Subsequent behavior depends on the authentication result: • If authentication succeeds, authentication continues down the LoginModule list. • If authentication fails, control returns immediately to the application (authentication does not proceed down the LoginModule list).
Sufficient	The LoginModule is not required. Subsequent behavior depends on the authentication result: • If authentication succeeds, control returns immediately to the application (authentication does not proceed down the LoginModule list). • If authentication fails, authentication continues down the LoginModule list.

Control Flag Value	Description
Optional (default)	The LoginModule is not required. Regardless of success or failure, authentication proceeds down the LoginModule list.

Example

Providers are listed in this order and with these controlFlag:

1. CertificateAuthenticationLoginModule (sufficient)
2. LDAP (optional)
3. NativeOS (sufficient)

A client doing certificate authentication (for example, X.509 SSO to SAP) can authenticate immediately. Subsequent modules are not called, because they are not required. If there are regular user name and password credentials, they go to LDAP, which may authenticate them, and set them up with roles from the LDAP groups they belong to. Then NativeOS is invoked, and if that succeeds, Sybase Unwired Platform picks up roles based on the Windows groups they are in.

Stacking LoginModules in SSO Configurations

(Not applicable to Online Data Proxy) Use LoginModule stacking to enable role-based authorization for MBOs and data change notifications (DCNs).

1. *Retrieving Roles for Subjects Authenticating to Single Sign-on Enabled Login Modules*

   The CertificateAuthenticationLoginModule does not extract role information. If MBOs and MBO operations have roles assigned, stack login modules to get roles for the user.
2. *Stacking Providers for DCN SSO Authentication*

   (Applies only to DCN events) Stack DCN providers with SSO providers to authenticate the SUP DCN User logical role with the SSO mechanisms. The users must be authenicated before they can be authorized.

Retrieving Roles for Subjects Authenticating to Single Sign-on Enabled Login Modules

The CertificateAuthenticationLoginModule does not extract role information. If MBOs and MBO operations have roles assigned, stack login modules to get roles for the user.

1. HttpAuthenticationLoginModule – username and password credentials are supplied by the user. If these credentials go to an LDAP/AD EIS, add an LDAPAuthorizer with appropriate properties to look up the LDAP subject and retrieve LDAP groups as roles. You can also use the csi-userrole authorizer; but role-mapping maintenance is onerous with a large user base.
2. CertificateAuthenticationLoginModule – use the csi-userrole provider to map logical roles to physical roles named user:*subject* where *subject* matches the common name (CN=*xxx*) from the X.509 certificate.

See *LDAP Configuration Properties* in *Sybase Control Center for Sybase Unwired Platform*.

Stacking Providers for DCN SSO Authentication

(Applies only to DCN events) Stack DCN providers with SSO providers to authenticate the SUP DCN User logical role with the SSO mechanisms. The users must be authenicated before they can be authorized.

The providers you use for SSO can vary.

1. For HttpAuthenticationLoginModule SSOimplementations, the CertificateAuthenticationLoginModule must be first in the list with the controlFlag set to "sufficient". If authentication succeeds, no other modules are used unless their controlFlags are set to "required".
2. For CertificateAuthenticationLoginModule implementations, , stack the chose DCN provider after the CertificateAuthenticationLoginModule and:
   a) Set the CertificateAuthenticationLoginModule's controlFlag to "sufficient", and order it first in the stack. This sequence allows normal device users to authenticate quickly.
   b) Choose any other user name and password-based login module to stack with its controlFlag set to either "optional" or "sufficient".

Security Provider Issues

If you experience problems with security configurations or the authentication or authorization providers in these configurations, check the Unwired Server logs for issues.

If no errors are being reported, despite failures that may occur while authenticating or authorizing users, you may need to increase the severity level of your logs. Search for *Logs* in the *Sybase Control Center for Sybase Unwired Platform*. If you are still experiencing issues, look for problems and solutions in the *Troubleshooting* guide.

Encrypting Synchronization for Replication Payloads

(Not applicable to Online Data Proxy) By default, the Unwired Server replication listener is configured to use TLS for end-to-end encryption (E2EE) on HTTP and HTTPS ports, and SSL for encryption on HTTPS ports.

If you do not require both TLS and SSL, you can disable either of them by modifying the replication synchronization listener in Sybase Control Center.

Once the listener is configured, applications must then connect to the Relay Server port and use an appropriate protocol:

- The administrator can define an application template in Sybase Control Center.
- The developer can call the Object API to set the E2EE and HTTPS items in the synchronization profile.

When applications are activated, clients receive their initial configuration settings from the application template and public keys, and HTTPS public certificate files are provisioned as part of these configuration settings.

1. *Changing Installed Certificates Used for Encryption*

   Unwired Server includes default certificates for all listeners. Since all installations use the same certificates by default, you must change these certificates with production-ready ones after you install Unwired Platform.
2. *Modifying Default Synchronization Listener Properties with Production Values*

   Once you have determined the degree of secure communication you require, you may need to modify default synchronization listener property values to disable one or more ports or synchronization protocols.
3. *Encryption Postrequisites*

   With the replication synchronization listener configured, ensure that the client application is provisioned with required artifacts and has connections configured accordingly.

See also

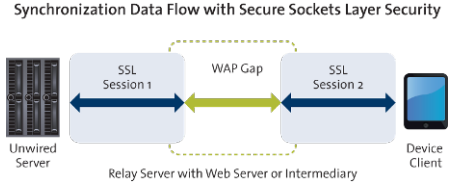
- *Provisioning Security Artifacts* on page 131
- *Unwired Server and Device Application Communications* on page 4
- *Certificate Creation (createcert) Utility* on page 184
- *Key Creation (createkey) Utility* on page 186
- *End-to-End Encryption with TLS* on page 102

End-to-End Encryption with TLS

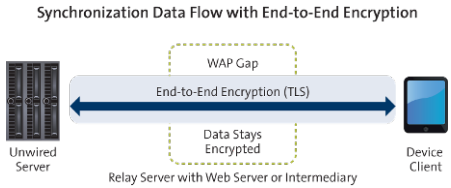
Wireless Application Protocol (WAP) has an issue commonly referred to as the WAP gap. You can Secure client/server synchronization with transport-layer security (TLS) to prevent WAP gap compromises.

TLS takes advantage of digital certificates and public-key cryptography to enable encryption, tamper detection, and certificate-based authentication for entity state replication environments.

A WAP gap breaks end-to-end communication data privacy. TLS encryption closes the WAP gap to ensure the synchronization stream is protected and secure. This diagram shows a traditional SSL synchronization stream passing through the WAP gap, where one SSL session encrypts the device-to-Relay Server stream and the other encrypts the Relay Server-to-Unwired Server stream:



Once you enable TLS to encrypt the entire synchronization data stream, you avoid passing unencrypted data through this WAP gap:



Note: End-to-end encryption for Unwired Platform supports RSA encryption only.

See also

- *Encrypting Synchronization for Replication Payloads* on page 101

Changing Installed Certificates Used for Encryption

Unwired Server includes default certificates for all listeners. Since all installations use the same certificates by default, you must change these certificates with production-ready ones after you install Unwired Platform.

TLS/SSL/HTTPS all use default certificates that require changing. Different listeners require different tools.

- Use **keytool** to manage certificates for the encryption of DCN, OData, and DOE listeners. These listeners all use the key and truststores (`keysstore.jks`), because these listeners require mutual certificate authentication. OCSP is only used for these listeners.
- Use **creatcert** to manage certificates for replication encryption. OCSP is not supported for replication.

Irrespective of the tool used, you can follow these general steps.

1. Generate new production-ready certificates:
   - If you use a PKI system, ensure that the generated certificates and key pairs are signed by the certificate authority (CA) certificate that is widely trusted in your organization. Unwired Platform is compliant with certificates and key pairs generated from most well-known PKI systems. Sybase recommends that you use this option.
   - If you do not use a PKI system, use the **keytool** or **createcert** utility to generate new self-signed certificates.
2. Import production-ready certificates, then update the security profile to associate these files with the Unwired Server encrypted port.
   a) Use the appropriate tool to import the new production certificates into the primary Unwired Server keystore, if that listener requires it.
   b) Configure the listener properties.

c) (Optional) If you are using a PKI system that includes OCSP and OCSP can be used by the listener, configure an OCSP responder. See *Enabling OCSP*.

See also

- *Certificate Creation (createcert) Utility* on page 184
- *Key Creation (createkey) Utility* on page 186

Modifying Default Synchronization Listener Properties with Production Values

Once you have determined the degree of secure communication you require, you may need to modify default synchronization listener property values to disable one or more ports or synchronization protocols.

Prerequisites

Ensure you have reviewed *Understanding Encyrption Requirements and Limitations*, and know what degree of secured or unsecured synchronization you require.

Task

For complete details on any of these properties, see *Configuring a Replication Listener* in the *Sybase Control Center for Sybase Unwired Platform*.

1. Open Sybase Control Center.
2. In the left navigation pane, select **Configuration** .
3. In the right administration pane, click the **General** tab.
4. Select **Replication** and click **Properties**.
5. Modify the protocol and port values:
   - To disable the HTTP port, unselect **Port**. Disabling this port means that you do not plan to use an unencrypted port, or use TLS for E2EE on this port (if you also disable all properties in step 9).

     Note: Do not unselect the HTTP port if you are using Relay Server. The RSOE cannot use the HTTPS port.
   - To change the default port value, delete port 2480 and enter a new value.
   - To disable the HTTPS , unselect **Secure port**. Disabling this port means that you do not plan to use HTTPS with SSL.
   - To change the default secure port value, delete port 2481 and enter a new value.

   Note: You cannot disable both ports.
6. To change any default HTTPS with SSL properties (particularly to set new values for production-ready certificates for HTTPS), modify these properties:

- Secure Sync Port Certificate – identifies the location of the security certificate used to encrypt and decrypt data transferred using SSL.
- Secure Sync Port Certificate Password – is used to decrypt the private certificate listed in the certificate file. You specify this password when you create the server certificate.
- Secure Sync Port Public Certificate – specify the file containing the public key that acts as the identity file for the synchronization port.
- Trusted Relay Server Certificate – if the Relay Server trusted certificate is configured, identifies the public security certificate location.

Note: If you have disabled the secure port, you do not need to configure these values.

7. To change any default E2EE properties (particularly to set new values for production ready certificates for E2EE), modify these properties:

   - E2E Encryption Certificate – specify the file containing the private key that acts as the identity file for Unwired Server.
   - E2E Encryption Certificate Password – set the password to unlock the encryption certificate.
   - E2E Encryption Public Key – specify the file containing the public key for Unwired Server.
   - E2E Encryption Type – specify the asymmetric cipher used for key exchange for end-to-end encryption. You can only use RSA encryption.

   Note: Leave E2EE values blank to disable end-to-end encryption.

Encryption Postrequisites

With the replication synchronization listener configured, ensure that the client application is provisioned with required artifacts and has connections configured accordingly.

Encrypting Other Listeners for Unwired Server

By default all other Unwired Platform listeners are encrypted using SSL. However, if you need to modify this configuration, review these steps.

1. *Changing Installed Certificates Used for Encryption*

   Unwired Server includes default certificates for all listeners. Since all installations use the same certificates by default, you must change these certificates with production-ready ones after you install Unwired Platform.

2. *Changing Keystore and Truststore Passwords*

   The Sybase Unwired Platform (used by both Unwired Server and Sybase Control Center to manage certificates and keys) keystore and truststore locations are protected by a password. In production environments, replacing default passwords is encouraged.

3. *Defining Certificates for SSL Encryption*

   Specify keystore and truststore certificates to be used for SSL encryption of Unwired Server communication ports. All security profiles use the same keystore and truststore.

4. *Creating an SSL Security Profile in Sybase Control Center*

   Security profiles define the security characteristics of a client/server session. Assign a security profile to a listener, which is configured as a port that accepts client connection requests of various protocols. Unwired Server uses multiple listeners. Clients that support the same characteristics can communicate to Unwired Server via the same port defined in the listener.

5. *Enabling OCSP*

   (Optional) Enable OCSP (Online Certificate Status Protocol) to determine the status of a certificate used to authenticate a subject: current, expired, or unknown. OCSP configuration is enabled as part of cluster level SSL configuration. OCSP checking must be enabled if you are using the CertificateAuthenticationLoginModule and have set Enable revocation checking to true.

See also

- *Encrypting Synchronization for Replication Payloads* on page 101

Changing Keystore and Truststore Passwords

The Sybase Unwired Platform (used by both Unwired Server and Sybase Control Center to manage certificates and keys) keystore and truststore locations are protected by a password. In production environments, replacing default passwords is encouraged.

Prerequisites

Before you begin, back up the contents of *SUP_HOME*`\Servers\UnwiredServer\Repository`.

Task

In production environments, use the keytool utility to change the default passwords for the keystore and truststore locations.

1. Open a command prompt window from this location: *SUP_HOME*`\Servers\UnwiredServer\Repository\Security`.
2. Run commands to change the current password for the keystore, truststore, and private key entries as required for your environment.

   You must enter the same password for a keystore and each of the private entries associated with that store.

   There is no provision in Sybase Control Center to specify a different password for the private key aliases.

For the keystore password, use: `keytool -storepasswd -new NewPwd -keystore Security\keystore.jks`

For the truststore password, use: `keytool -storepasswd -new NewPwd -truststore Security\truststore.jks`

For private key entries in keystore, use: `keytool -keypasswd -alias Name -new NewPwd -keystore Security\keystore.jks`

3. At the prompt, enter the current password.

   If this is the first time changing the password, enter the default password of `changeit`. Otherwise, enter the current password.

4. In Sybase Control Center, configure the SSL certificates to use these passwords. If these certificates are already configured, update the passwords currently configured.

   Click **Configuration** > **General**, then click the **SSL Configuration** tab. For details, see *Defining Certificates for SSL Encryption*.

   If you do not ensure the correct password is set, you can expect a connection failure. See *Key or Keystore Messages Received in Server Log* in the *Troubleshooting* guide.

5. Restart all Sybase Unwired Platform services using the Windows Control Panel services tool.

See also

- *Changing Installed Certificates Used for Encryption* on page 103

Defining Certificates for SSL Encryption

Specify keystore and truststore certificates to be used for SSL encryption of Unwired Server communication ports. All security profiles use the same keystore and truststore.

1. In the left navigation pane, select **Configuration**
2. In the right administration pane, select the **General** tab.
3. From the menu bar, select **SSL Configuration**.
4. To configure SSL encryption for all security profiles, complete these fields:
   - **Keystore Location –** the relative path name indicating the location where the keys and certificates are stored. Certificates used for administration and data change notification ports are stored in the keystore. The path should be relative to *SUP_HOME*`\Servers\UnwiredServer`.
   - **Keystore Password –** the password that secures the key store.
   - **Truststore Location –** therelative path name for the public key certificate storage file. The Certificate Authority (CA) certificates used to sign certificates store their public keys in the truststore. The path should be relative to *SUP_HOME*`\Servers\UnwiredServer`.
   - **Truststore Password –** the password that secures the truststore.

Note: If at any point you have changed the password for the keystore and truststore with keytool, then you must remember to update the password here as well. The password must be used with all aliases as well. To update the alias, use a command similar to this one:

```
keytool -keypasswd -alias sample1 -keypass changeit -new
changeit2 -keystore keystore.jks
```

```
keytool -keypasswd -alias sample2 -keypass changeit -new
changeit2 -keystore keystore.jks
```

5. Click **Save**.

Next

Create an SSL security profile that uses the selected certificates.

Creating an SSL Security Profile in Sybase Control Center

Security profiles define the security characteristics of a client/server session. Assign a security profile to a listener, which is configured as a port that accepts client connection requests of various protocols. Unwired Server uses multiple listeners. Clients that support the same characteristics can communicate to Unwired Server via the same port defined in the listener.

Note: A security profile can be used by one or more servers in a cluster, but cannot used by multiple clusters.

1. In the left navigation pane, select **Configuration**
2. In the right administration pane, select the **General** tab.
3. From the menu bar, select **SSL Configuration**.
4. In the **Configure security profile table**:
   a) Enter a name for the security profile.
   b) Enter a certificate alias. This is the alias of a key entry in the keystore. Make sure the key password of this key entry is the same as the keystore password.
   c) Select an authentication level:

      If the security profile authenticates only the server, then only the server must provide a certificate to be accepted or rejected by the client. If the security profile authenticates both the client and the server, then the client is also required to authenticate using a certificate; both the client and server will provide a digital certificate to be accepted or rejected by the other.

Profile	Authenticates	Cipher suite(s)
intl	server	• SA_EXPORT_WITH_RC4_40_MD5 • RSA_EXPORT_WITH_DES40_CBC_SHA

Profile	Authenticates	Cipher suite(s)
intl_mutual	client/server	• RSA_EXPORT_WITH_RC4_40_MD5 • RSA_EXPORT_WITH_DES40_CBC_SHA
simple	server	RSA_WITH_NULL_MD5 RSA_WITH_NULL_SHA
simple_mutual	client/server	RSA_WITH_NULL_MD5 RSA_WITH_NULL_SHA
strong	server	• RSA_WITH_3DES_EDE_CBC_SHA • RSA_WITH_RC4_128_MD5 • RSA_WITH_RC4_128_SHA
strong_mutual	client/server For example, this is the required option for mutual authentication of Unwired Platform and Gateway.	• RSA_WITH_3DES_EDE_CBC_SHA • RSA_WITH_RC4_128_MD5 • RSA_WITH_RC4_128_SHA
domestic	server	• RSA_WITH_3DES_EDE_CBC_SHA • RSA_WITH_RC4_128_MD5 • RSA_WITH_RC4_128_SHA • RSA_WITH_DES_CBC_SHA • RSA_EXPORT_WITH_RC4_40_MD5 • RSA_EXPORT_WITH_DES40_CBC_SHA • TLS_RSA_WITH_NULL_MD5 • TLS_RSA_WITH_NULL_SHA

Profile	Authenticates	Cipher suite(s)
domestic_mutual	client/server	• RSA_WITH_3DES_EDE_CBC_SHA • RSA_WITH_RC4_128_MD5 • RSA_WITH_RC4_128_SHA • RSA_WITH_DES_CBC_SHA • RSA_EXPORT_WITH_RC4_40_MD5 • RSA_EXPORT_WITH_DES40_CBC_SHA • RSA_WITH_NULL_MD5 • RSA_WITH_NULL_SHA

5. Click **Save**.
6. From the **Components** menu, assign the security profile to the desired management or communication ports.

Enabling OCSP

(Optional) Enable OCSP (Online Certificate Status Protocol) to determine the status of a certificate used to authenticate a subject: current, expired, or unknown. OCSP configuration is enabled as part of cluster level SSL configuration. OCSP checking must be enabled if you are using the CertificateAuthenticationLoginModule and have set Enable revocation checking to true.

Enable OCSP for a cluster when configuring SSL.

1. In the left navigation pane, select **Configuration**.
2. In the right administration pane, select the **General** tab.
3. From the menu bar, select **SSL Configuration**.
4. To enable OCSP when doing certificate revocation checking, check **Enable OCSP**.
5. Configure the responder properties (location and certificate information):

Responder Property	Details
URL	A URL to responder, including its port. For example, `https://ocsp.example.net:80`.

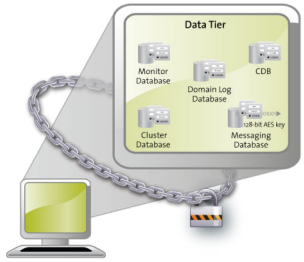
Responder Property	Details
Certificate subject name	The subject name of the responder's certificate. By default, the certificate of the OCSP responder is that of the issuer of the certificate being validated. Its value is a string distinguished name (defined in RFC 2253), which identifies a certificate in the set of certificates supplied during cert path validation. If the subject name alone is not sufficient to uniquely identify the certificate, the subject value and serial number properties must be used instead. When the certificate subject name is set, the certificate issuer name and certificate serial number are ignored. For example, `CN=MyEnterprise, O=XYZCorp`.
Certificate issuer name	The issuer name of the responder certificate. For example, `CN=OCSP Responder, O=XYZCorp`.
Certificate serial number	The serial number of the responder certificate.

See also

- *Creating an SSL Security Profile in Sybase Control Center* on page 108

CHAPTER 4 Data Tier Security

The data tier consists of multiple databases, each of which plays a unique role in Unwired Platform, and contains various types of sensitive data that must be secured.



Securing the Data Infrastructure

Secure data by first protecting the infrastructure on which it resides, then securing runtime databases.

1. *Setting File System Permissions*

   Unwired Platform runs as a collection of Windows services. During installation, you are prompted with a Logon as request. The credentials collected are then used to run the service under that account. In a cluster installation, the same Windows user would be configured for all installations and respective Window services that are subsequently installed.
2. *Securing Backup Artifacts*

   If you perform backups of Unwired Platform, you should also secure the backup artifacts.

Setting File System Permissions

Unwired Platform runs as a collection of Windows services. During installation, you are prompted with a **Logon as** request. The credentials collected are then used to run the service under that account. In a cluster installation, the same Windows user would be configured for all installations and respective Window services that are subsequently installed.

You can restrict permissions after installation by removing most users and groups from the Unwired Platform installation directory.

1. Open File Explorer.

2. Right-click *SUP_HOME*, and click **Properties**.
3. On the **Security** tab, click **Advanced**.
4. Unselect **Inherit from parent the permission entries that apply to child objects. Include these with entries explicitly defined here.**.
5. In the confirmation pop-up, choose **Copy**, then select **Replace permission entries on all child objects with entries shown here that apply to child objects.**.
6. In the table of Permission entries, remove all users except the user account that was configured as the Logon user for the Windows services. If another user is responsible for some activities extend the necessary permissions to this administrator. For example, if the individual is only reading log files, you may choose to limit permissions to read only.
7. Click **OK**.

Securing Backup Artifacts

If you perform backups of Unwired Platform, you should also secure the backup artifacts.

1. Protect backups with Administrator and SYSTEM permissions.
2. Ensure that role-based access to the backup folder uses the same permissions model used for the production servers. The same IT users that can access the production database folders should be the same ones that can accessing the backup folders.
3. Perform any additional enterprise security requirements.

Securing Data Tier Databases

Secure all databases installed as the Unwired Platform data tier. You can change DBA passwords, grant DBA permissions to other users, and encrypt data and logs.

See also

- *Securing the Data Infrastructure* on page 113
- *Encrypting Device Data* on page 126

Changing DBA Passwords for SQLAnywhere Databases in a Cluster Deployment

By default, Unwired Platform uses multiple SQLAnywhere databases to support the server runtime environment and transactions. Unwired Server accesses these databases with the DBA user identity.

During installation, enter a custom password for the DBA user on each of the SQLAnywhere databases used by the runtime: the cache database (CDB), the cluster database, the log database, and monitor database.

In a cluster deployment, these servers are used:

- The monitor and domain log databases use `LogDataDB`.
- The cache database uses `CacheDB`.
- The cluster database uses `ClusterDB`.

1. Stop all instances of Unwired Server, as well as all database services.

   For a list of database services, search for *Unwired Platform Windows Services* in *System Administration*.
2. On data tier host:
   a) Set the location of the BIN directory for your operating system (32-bit or 64-bit):

      ```
      set SQLANY12_BIN=SUP_HOME\Servers\SQLAnywhere12\bin<32|64>
      ```

   b) Set the path to the data:

      If you are using a single node, run:

      ```
      set DATA_PATH= SUP_HOME\Servers\UnwiredServer\data
      ```

      If you are using a cluster deployment, run:

      ```
      set DATA_PATH= SUP_HOME\data\CDB
      ```
3. Use **dbisql** to change passwords for each database as required:

   For the cache database, use:

   ```
   "%SQLANY12_BIN%\dbisqlc" -q -c "Server=default;DBF=%DATA_PATH%
   \default.db;
   UID=DBA;PWD=ExistingPwd" grant connect to dba identified by NewPwd
   ```

   For the cluster database, use:

   ```
   "%SQLANY12_BIN%\dbisqlc" -q -c "Server=clusterdb;DBF=%DATA_PATH%
   \clusterdb.db;
   UID=DBA;PWD=ExistingPwd" grant connect to dba identified by NewPwd
   ```

   For the monitor database, use:

   ```
   "%SQLANY12_BIN%\dbisqlc" -q -c "Server=monitordb;DBF=%DATA_PATH%
   \monitordb.db;
   UID=DBA;PWD=ExistingPwd" grant connect to dba identified by NewPwd
   ```

   For the domain log database, use:

   ```
   "%SQLANY12_BIN%\dbisqlc" -q -c "Server=domainlogdb;DBF=%DATA_PATH
   %\domainlogdb.db;
   UID=DBA;PWD=ExistingPwd" grant connect to dba identified by NewPwd
   ```
4. To register the change with the runtime, run this command on each Unwired Server node host:

   ```
   Register-dsn.bat cdb.install_type cdb.serverhost cdb.serverport
   cdb.username
    %CDB_PASSWORD% cdb.servername cldb.dsnname %CLDB_PASSWORD%
   %MONITORDB_PASSWORD% %DOMAINLOGDB_PASSWORD%
   ```

To see the values used in the properties of this command, open the *SUP_HOME*\Servers\UnwiredServer\Repository\Instance\com\sybase\sup\server\SUPServer\sup.properties file and search for the corresponding property.

5. If you receive an `Invalid user ID` or `Invalid password` error, you may have already changed the password from "sql" to different one). In this case:
   a) Backup **register-dsn.bat**.
   b) Open this file in a text editor and locate:

```
IF "default" == "%CDB_INSTALLTYPE%" (
    echo Changing DBA password for databases ...
    "%SQLANY12_BIN%\dbisqlc" -q -c "Server=default;DBF=
%DJC_HOME%\data\default.db;UID=DBA;PWD=sql" grant connect to
dba identified by %CDB_PASSWORD%
    "%SQLANY12_BIN%\dbisqlc" -q -c "Server=clusterdb;DBF=
%DJC_HOME%\data\clusterdb.db;UID=DBA;PWD=sql" grant connect to
dba identified by %CLDB_PASSWORD%
    "%SQLANY12_BIN%\dbisqlc" -q -c "Server=monitordb;DBF=
%DJC_HOME%\data\monitordb.db;UID=DBA;PWD=sql" grant connect to
dba identified by %MONITORDB_PASSWORD%
    "%SQLANY12_BIN%\dbisqlc" -q -c "Server=domainlogdb;DBF=
%DJC_HOME%\data\domainlogdb.db;UID=DBA;PWD=sql" grant connect
to dba identified by %DOMAINLOGDB_PASSWORD%
)
```

   c) Replace `PWD=sql` with the `PWD=`*`PreviousPassword`*.
6. To register the change with the synchronization server, run:

```
run-ant-config.bat configure-mlsrv.ini configure-sup -
Dsqlany12.bin=%SQLANY12_BIN%
```

   where *%SQLANY12_BIN%* is substituted with the path value recorded in the `asa-setenv.bat`.
7. Execute:

```
updateProps -u cdb.username -p NEWPwd -d cldb.dsnname -nv
"cdb.password=NEWPwd#cldb.password=NEWPwd#monitoringdb.password=N
EWPwd#domainlogdb.password=NEWPwd"
```

8. Restart all database services, then all Unwired Servers.

Changing DBA Passwords for SQLAnywhere Databases in a Single-Node Installation

By default, Unwired Platform uses multiple SQLAnywhere databases to support the server runtime environment and transactions. Unwired Server accesses these databases with the DBA user identity.

During installation, enter a custom password for the DBA user on each of the SQLAnywhere databases used by the runtime: the cache database (CDB), the cluster database, the log database, and monitor database.

In single-node deployment, a single database server named `CacheDB` supports all installed databases.

1. Stop all instances of Unwired Server, as well as all database services.
   For a list of database services, search for *Unwired Platform Windows Services* in *System Administration*.
2. On data tier host:
   a) Set the location of the BIN directory for your operating system (32-bit or 64-bit):
      ```
      set SQLANY12_BIN=SUP_HOME\Servers\SQLAnywhere12\bin<32|64>
      ```
   b) Set the path to the data:
      If you are using a single node, run:
      ```
      set DATA_PATH= SUP_HOME\Servers\UnwiredServer\data
      ```
      If you are using a cluster deployment, run:
      ```
      set DATA_PATH= SUP_HOME\data\CDB
      ```
3. Use **dbisql** to change passwords for each database as required:
   For the cache database, use:
   ```
   "%SQLANY12_BIN%\dbisqlc" -q -c "Server=default;DBF=%DATA_PATH%
   \default.db;
   UID=DBA;PWD=ExistingPwd" grant connect to dba identified by NewPwd
   ```
   For the cluster database, use:
   ```
   "%SQLANY12_BIN%\dbisqlc" -q -c "Server=clusterdb;DBF=%DATA_PATH%
   \clusterdb.db;
   UID=DBA;PWD=ExistingPwd" grant connect to dba identified by NewPwd
   ```
   For the monitor database, use:
   ```
   "%SQLANY12_BIN%\dbisqlc" -q -c "Server=monitordb;DBF=%DATA_PATH%
   \monitordb.db;
   UID=DBA;PWD=ExistingPwd" grant connect to dba identified by NewPwd
   ```
   For the domain log database, use:
   ```
   "%SQLANY12_BIN%\dbisqlc" -q -c "Server=domainlogdb;DBF=%DATA_PATH
   %\domainlogdb.db;
   UID=DBA;PWD=ExistingPwd" grant connect to dba identified by NewPwd
   ```
4. To register the change with the runtime, run this command:
   ```
   Register-dsn.bat cdb.install_type cdb.serverhost cdb.serverport
   cdb.username
    %CDB_PASSWORD% cdb.servername cldb.dsnname %CLDB_PASSWORD%
   %MONITORDB_PASSWORD% %DOMAINLOGDB_PASSWORD%
   ```
   To see the values used in the properties of this command, open the *SUP_HOME*\Servers\UnwiredServer\Repository\Instance\com\sybase\sup\server\SUPServer\sup.properties file and search for the corresponding property.
5. If you receive an `Invalid user ID` or `Invalid password:`
   a) Backup **register-dsn.bat**.
   b) Open this file in a text editor and locate:
      ```
      IF "default" == "%CDB_INSTALLTYPE%" (
          echo Changing DBA password for databases ...
      ```

```
    "%SQLANY12_BIN%\dbisqlc" -q -c "Server=default;DBF=
%DJC_HOME%\data\default.db;UID=DBA;PWD=sql" grant connect to
dba identified by %CDB_PASSWORD%
    "%SQLANY12_BIN%\dbisqlc" -q -c "Server=clusterdb;DBF=
%DJC_HOME%\data\clusterdb.db;UID=DBA;PWD=sql" grant connect to
dba identified by %CLDB_PASSWORD%
    "%SQLANY12_BIN%\dbisqlc" -q -c "Server=monitordb;DBF=
%DJC_HOME%\data\monitordb.db;UID=DBA;PWD=sql" grant connect to
dba identified by %MONITORDB_PASSWORD%
    "%SQLANY12_BIN%\dbisqlc" -q -c "Server=domainlogdb;DBF=
%DJC_HOME%\data\domainlogdb.db;UID=DBA;PWD=sql" grant connect
to dba identified by %DOMAINLOGDB_PASSWORD%
)
```

c) Replace `PWD=sql` with the `PWD=`*`NewPassword`*.

6. To register the change with the synchronization server, run:

```
run-ant-config.bat configure-mlsrv.ini configure-sup -
Dsqlany12.bin=%SQLANY12_BIN%
```

where *%SQLANY12_BIN%* is substituted with the path value recorded in the `asa-setenv.bat`.

7. Execute:

```
updateProps -u cdb.username -p NEWPwd -d cldb.dsnname -nv
"cdb.password=NEWPwd#cldb.password=NEWPwd#monitoringdb.password=N
EWPwd#domainlogdb.password=NEWPwd"
```

8. Restart all database services, then all Unwired Servers.

Encrypting Data and Log Outputs

Database files and log files that are used as part of the Sybase Unwired Platform data tier can be encrypted. The databases that use this database type are the CDB, the monitoring database, and the domain log database.

1. Shut down the database server.
2. Stop all Sybase Unwired Platform services.
3. Navigate to `.../UnwiredServer/bin/sqlanywhereoptions.ini` to locate the required `*.db` file.
4. Launch dbisql from *SUP_HOME*`\Servers\SQLAnywhere`*XX*`\BIN`*XX*.
5. Connect to a database, other than the client database you want to encrypt.
6. From **dbisql**, issue:

```
CREATE ENCRYPTED DATABASE 'newdbfile' FROM 'existingdbfile' KEY
'someKey' ALGORITHM 'algorithm'
```

Supported algorithms include:

- SIMPLE
- AES
- AES256

- AES_FIPS
- AES256_FIPS

Note: FIPS options are available only as a separately licensed option for SQLAnywhere.

7. Edit `sqlanywhereoptions.ini` to:

```
SUP_HOME\Servers\UnwiredServer\bin\../data/default-enc.db" -ek
secret ...
```

8. Once the database files and log files are encrypted:
   a) Shut down the database server.
   b) Restart the database server with the -ek <encryption key> database option.
      - For a single node, use the -ek <encryption key> directly after the target `newdbfile` full path.
      - For a cluster node, you must change the target option file. Use the -ek <encryption key> directly after the target `newdbfile` full path as a database option.

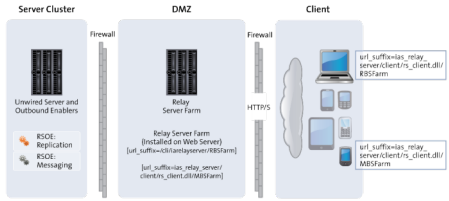
   This modifies the server startup to use the encrypted copy of the database file.
9. Restart all stopped services.

Note: If you use the Start Sybase Unwired Platform Services desktop shortcut, the `.ini` file is overwritten. Therefore, you should set the `.ini` file to as read only for the account that runs the database service and prohibit all access for any other accounts in order to keep the encryption key secret.

CHAPTER 5 DMZ Security

DMZ security involves controlling Internet traffic to private networks by installing a Relay Server between your inner and outer firewalls.

The outer firewall has HTTP and HTTPS ports open to allow client Internet traffic to reach the Relay Server.



Relay Server as Firewall Protection

The Relay Server is a pair of Web server plug-ins, which you can install on an Internet Information Service (IIS) server on Windows, or on the Apache Web server on Linux.

The Relay Server is intended to run between a company's inner and outer firewalls. The outer firewall has HTTP and HTTPS ports open to allow client Internet traffic to reach the Relay Server. The client's URL includes the address of the client-side plug-in of the Relay Server and the name of the back-end Sybase Unwired Platform "farm" the client is trying to reach. A farm includes multiple Relay Servers for load balancing and fault tolerance. The network administrator must install a load balancer in front of the Relay Servers. The load balancer is not included with Sybase Unwired Platform. To make the interaction secure, clients should use end-to-end encryption.

The server-side plug-in accepts connections from various Relay Server Outbound Enabler (RSOE) processes, which indicate to the Relay Server what back-end farm each process represents. The Relay Server matches the farm name in the client's request to a server-side plug-in connection, and routes the client's request contents to that connection. Other than the farm name in the request URL, the Relay Server knows nothing about the content of these messages. The clients are not authenticated or authorized in any way. The client information is in memory and therefore is not susceptible to interception or modification. But, if the administrator turns certain tracing options up very high, data may get copied to log files. If end-to-end encryption is used, the data is undecipherable.

Security administrators secure the Relay Server as they would with any other Web server or proxy server they run between firewalls, so the same security precautions should be taken of setting up a proxy server.

See also

- *RSOE as the Unwired Server Protection* on page 122
- *Relay Server and RSOE Communication Security* on page 122
- *Configuring Connection Properties for Relay Server Components* on page 123

RSOE as the Unwired Server Protection

One RSOE process is installed in each Unwired Server cluster member, in front of each synchronization subcomponent that communicates with a client. Replication service components and messaging service components both use RSOEs attached to their communication ports.

The RSOE configuration enables the Relay Server to identify the RSOE and connect to it. The RSOE configuration also has a single port number that enables the RSOE to make an `http://localhost:port` connection whenever a client request comes to it from the Relay Server.

See also

- *Relay Server as Firewall Protection* on page 121
- *Relay Server and RSOE Communication Security* on page 122
- *Configuring Connection Properties for Relay Server Components* on page 123

Relay Server and RSOE Communication Security

The RSOE runs on the same computer as an Unwired Server and is configured with the address of a Relay Server (the inner firewall is open to outgoing traffic, but not incoming traffic).

The RSOE connects to the Relay Server via HTTP or HTTPS and identifies itself through the Media Access Control (MAC) address, security token, and the back-end Sybase Unwired Platform farm it services. The Relay Server identifies the RSOE's authenticity. If Relay Server accepts the RSOE's identity, it sends RSOE a list of all other RSOEsin the Relay Server farm. The RSOE establishes a blocking GET HTTP request to eachfarm member. When a Relay Server receives a client request for a given Sybase Unwired Platform farm, it picks one of the available RSOE connections and sends the client request there.

In this way, the network administrator need not open inner firewall ports to allow connection requests into the intranet. All connection requests come from within the intranet. Avoiding firewall portholes protects the intranet from hackers who breach the outer firewall.

This network traffic contains exactly the same content, and thus the same security concerns as network communication between the device application or database and the Relay Server.

See also

- *Relay Server as Firewall Protection* on page 121
- *RSOE as the Unwired Server Protection* on page 122
- *Configuring Connection Properties for Relay Server Components* on page 123

Configuring Connection Properties for Relay Server Components

In most highly available deployments, you configure both Relay Server and RSOE to use HTTP when connecting to Unwired Server on the corporate LAN. In more specialized, less available deployments (for example, where BES is inside the corporate LAN and is configured to connect directly to Unwired Server without any load-balancing by Relay Server), use HTTPS.

Configure both Relay Server and Outbound Enablerconnections:

1. *Configuring Relay Server Connection Properties*

   Configure the connection type used by Relay Server to connect to the Unwired Server.
2. *Configuring Outbound Enabler Connection Properties*

   Outbound Enabler establishes two connections. You can configure connections from the Outbound Enabler to Relay Server to use either HTTP or HTTPS. However, connections from the Outbound Enabler to Unwired Server can only use HTTP, so this connection does not require configuration.

See also

- *Relay Server as Firewall Protection* on page 121
- *RSOE as the Unwired Server Protection* on page 122
- *Relay Server and RSOE Communication Security* on page 122

Configuring Relay Server Connection Properties

Configure the connection type used by Relay Server to connect to the Unwired Server.

Prerequisites

If you are using a load balancer, configure it with the same properties as Relay Server.

Task

1. In the navigation pane, click the Unwired Server cluster name.

2. In the administration pane, click the **Relay Servers** tab.
3. Click **New**.
4. When you reach the **General** properties page, configure these properties:
   - In highly available deployments where Relay Server is deployed to the DMZ, enable the HTTP port.
   - When Relay Server is installed on the corporate LAN, enable the HTTPS port.
5. Configure all remaining properties as documented in *Creating a Custom Relay Server Configuration* in the *Landscape Design and Integration* guide.

Configuring Outbound Enabler Connection Properties

Outbound Enabler establishes two connections. You can configure connections from the Outbound Enabler to Relay Server to use either HTTP or HTTPS. However, connections from the Outbound Enabler to Unwired Server can only use HTTP, so this connection does not require configuration.

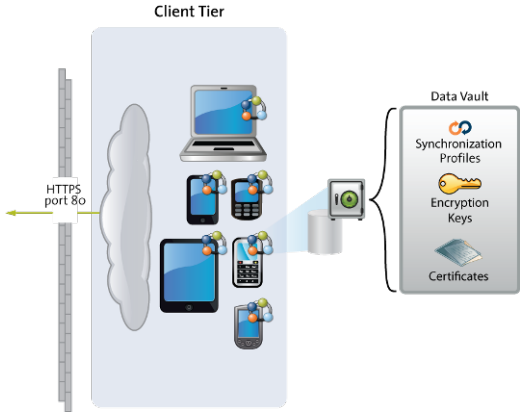
After you configure Relay Server, configure Outbound Enablers on each Unwired Server node.

1. In the navigation pane, click **Servers > <*ServerNode*> > Server Configuration**.
2. In the administration pane, select the **Outbound Enabler** tab, then click **New**.
3. In the General properties page, ensure you configure the Relay Server port for the type of connection you require (either HTTP or HTTPS).
4. If you choose HTTPS for the Outbound Enabler's client connection, either import the Relay Server certificate or that of the RelayServer's certificate signing CA into the *SUP_HOME*`\Servers\UnwiredServer\Repository\Security` directory on the primary Unwired Server.
5. Configure the Outbound Enabler's Certificate file, and optionally the Trusted certificate (when the certificate file contains multiple certificates) with appropriate values.
6. Configure all other values as described in *Configuring Relay Servers and Outbound Enablers* in the *Installation Guide for Runtime*.

CHAPTER 6 Device Security

You can combine multiple mechanisms to fully secure devices. In addition to using the built-in security features of both the device or Unwired Platform, Sybase recommends that you also use Afaria so you can remotely initiate security features as required.

Unwired Platform security features for devices include data encryption, login screens, and data vaults for storing sensitive data.



Limiting Application Access

Application access to Unwired Platform runtime is tightly controlled: before a user can access a mobile application, he or she must provide a passcode; before the application can access the runtime, the application must be registered and provisioned with required connections and security configurations.

Applications that do not require tight security can use anonymous access. Anonymous access applications can be run without a specific user name/authorization code or code/password.

1. *Encrypting Device Data*
   Encrypting all data on the device client requires multiple techniques.
2. *Registering Applications, Devices, and Users*
   Before any application can access the runtime, the user, device, and application must be identified by first registering with Unwired Server and pairing them with a device and user entry. Only when all three entities are known, can subscriptions can be made or data synchronized.
3. *Locking and Unlocking a Subscription*
   (Not applicable to Online Data Proxy) Create a subscription template to lock or unlock a subscription. A subscription determines what data set the device user receives upon

synchronization and how frequently the synchronization can occur. Lock the subscription to prevent modification to the template and control the synchronization frequency

4. *Locking and Unlocking Application Connections*

   Lock or unlock connections to control which users are allowed to synchronize data. Locking an application connection is an effective way to disable a specific user without making changes to the security profile configuration to which he or she belongs. Locking an application connection blocks delivery of generated data notifications to the replication-based synchronization clients.

Encrypting Device Data

Encrypting all data on the device client requires multiple techniques.

Some Unwired Platform components do not support encryption. Review this table to see which components can enable this security feature.

Component	Implementation Notes
Device data	Sybase recommends full device encryption with Afaria. See the Afaria documentation for details.
Device client database	(Not applicable to Online Data Proxy) A `<package>DB.generateEncryptionKey()` method in the Object API for MBO packages should always be used during application initialization. It computes a random AES-256 bit encryption key used to encrypt the client database. The encryption key is stored in the data vault.
Data vault	The DataVault APIs provide a secure way to persist and encrypt data on the device. The data vault uses AES-256 symmetric encryption of all its contents. The AES key is computed as a hash of the passcode provided and a "salt" value that can be supplied by the device application developer, or automatically generated through the API.

Registering Applications, Devices, and Users

Before any application can access the runtime, the user, device, and application must be identified by first registering with Unwired Server and pairing them with a device and user entry. Only when all three entities are known, can subscriptions can be made or data synchronized.

In Sybase Control Center, Platform administrators set up an application connection template for applications. Part of this template includes a property that enables automatic registration.

- When automatic registration is enabled, a device user need only provide valid Sybase Unwired Platform credentials that are defined as part of the security configuration. If the application connection template specifies a logical role, the user must have a physical role that maps to the logical role in order to access the application.

- When automatic registration is disabled, the platform administrator must provide the user a user name and passcode out-of-band. This is the passcode initially required by login screens to access the application for the first time, and expires within a predetermined time period (72 hours, by default).

Note: Choose to use automatic registrations carefully, especially if there are multiple application connection templates for the same application. The combined criteria of the application ID and security configuration used by the device application trigger a search for a matching template that is used to complete the automatic registration. However, if the security configuration is not sent by the device application, and the server finds multiple templates, registration fails.

See also

- *Locking and Unlocking a Subscription* on page 127

Registering Application Connections

Devices can be registered using either an activation code during the registration of the device or application, or to allow automatic registration.

When a package is deployed, an application connection template is automatically created. As long as the user is able to authenticate to the security configuration associated with the application, they are registered. If the application connection template specifies a logical role, the user must have a physical role that maps to the logical role to access the application. If automatic registration is disabled, the administrator must generate an activation code. For details, see *Mobile Application Life Cycle* and search for *Manual Connection Registration with Activation Codes*.

1. Select the application connection template, and click **Properties**.
2. Navigate to the **Application Settings** tab and set the **Automatic Registration Enabled** property to True or False. If True, no activation code is required.

Defining Applications

Applications are recognized by Unwired Server by the properties that define them. Administrators define applications with a unique application ID and other key application properties, such as domain, packages, security configuration, and connection templates.

An application cannot register a connection unless a definition has been created for it. If your development team has not yet set these application properties, administrators must do so before the application connection can be registered.

Locking and Unlocking a Subscription

(Not applicable to Online Data Proxy) Create a subscription template to lock or unlock a subscription. A subscription determines what data set the device user receives upon

synchronization and how frequently the synchronization can occur. Lock the subscription to prevent modification to the template and control the synchronization frequency

1. In the left navigation pane, expand the **Packages** folder and select the replication-based package to configure.
2. In the right administration pane, click the **Subscriptions** tab.
3. From the menu bar, select **Templates**, then click **New**.
4. Select **Admin Lock** to prevent device users from modifying the push synchronization state or sync interval value configured in the subscription. If the admin lock is disabled, the device client user can change these properties, and these changes take effect the next time the client user synchronizes the package to which the subscription applies.

See also

- *Registering Applications, Devices, and Users* on page 126

Locking and Unlocking Application Connections

Lock or unlock connections to control which users are allowed to synchronize data. Locking an application connection is an effective way to disable a specific user without making changes to the security profile configuration to which he or she belongs. Locking an application connection blocks delivery of generated data notifications to the replication-based synchronization clients.

1. In the left navigation pane, select the **Applications** node.
2. In the right administration pane, select the **Application Connections** tab.
3. Select the application connection you want to manage, and:
   - If the connection is currently unlocked and you want to disable synchronization, click **Lock**.
   - If the connection is currently locked and you want to enable synchronization, click **Unlock**.
4. In the confirmation dialog, click **OK**.

Securing Sensitive Data On-Device with DataVault

(Not applicable to Hybrid Web Container) Developers should use a data vault with device applications to securely store “secrets” on the device. Data vaults are added using the DataVault API.

The data vault holds sensitive artifacts securely, because all data or artifacts in the data vault is strongly encrypted with an AES-256 bit key. Contents can include encryption keys, user and application login credentials, sync profile settings, certificates (as BLOBS).

The data vault requires a password to unlock and access the data from the application. Therefore, a device application must prompt the user to enter this password when the

application is opened. Once unlocked, the application can retrieve any other secrets from the vault as needed, all without prompting the user.

Administrators can define a password policy through Sybase Control Center that defines the requirements for an acceptable password. The password policy is stored in the server-side settings database and the client gets those settings when it connects to Unwired Server as part of the settings exchange protocol.

When the client receives the password policy settings, it can populate the settings objects to the data vault. The datavault stores the settings. The client uses the DataVault API to create a vault with a default password, set the password policy, and change the password to a password compatible with the policy. If the password is not changed after setting a password policy, the application will throw an exception if you then try to access the application or unlock the vault with an incompatible password.

Administrators should discuss the data vault strategy before it is implemented, especially regarding:

- **Failed logins –** Developers can set the number of failed login attempts allowed before the data vault is deleted. Once the vault is deleted, the encrypted databases are not useable. The application needs to be re-installed, or re-initialized, including deleting the database files to recover.
- **Timeouts –** Developers can also set a timeout value so that the data vault locks itself when it's not in use. The user must re-enter the vault password to resume using the application.

For more details about the data vault, see *Data Vault* in the developer guide for your application type.

1. *Enabling and Configuring a Password Policy for Data Vault Logins*

   Administrators can create a password policy for device application logins in a new or existing application connection template. A password policy ensures that user-defined passwords conform to corporate security requirements.
2. *Using Login Screens for Data Vaults*

   An application that implements a login screen is considered to be secure. Mobile application developers are responsible for creating login screens for the applications they create. A login screen allows the device user to enter a passcode to unlock the data vault.

See also

- *Using Login Screens for Data Vaults* on page 130

Enabling and Configuring a Password Policy for Data Vault Logins

Administrators can create a password policy for device application logins in a new or existing application connection template. A password policy ensures that user-defined passwords conform to corporate security requirements.

A policy cannot be enforced unless developers add enforcement code to the data vault for an application. For information about creating a data vault that enforces the platform policy, see *Creating a DataVault that Enforces Password Policy*.

1. In the left navigation pane, click the **Applications** node.
2. In the right administration pane, click the **Application Connection Templates** tab.
3. Choose one of the following:
   - To create a new template with a password policy, click **New**.
   - To edit an existing template, select the name of the template and click **Properties**.
4. In the navigation pane, select the **Password Policy** category
5. In the right pane, configure the properties of the password policy.
6. Click **OK**.

Using Login Screens for Data Vaults

An application that implements a login screen is considered to be secure. Mobile application developers are responsible for creating login screens for the applications they create. A login screen allows the device user to enter a passcode to unlock the data vault.

A secure application that uses a login screen:

1. Prompts the user to enter the datavault passcode to open the application and get access to the local client database. If the wrong passcode is used, the application is rendered useless: the key that encrypts and decrypts data in the vault cannot be used to access data until this code is accurately entered.

   After a certain amount of time passes, the login in screen can be redeployed to prompt the user to re-enter the passcode.
2. Can be locked out after a configured number of retries.
3. Can self-destruct after a set number of incorrect passcode attempts.

   When this occurs, the device user must uninstall, reinstall, then perform an initial synchronization to recover from a destroyed data vault.

To implement a login screen you must create the login and the define the password. The screen and the password unlock the DataVault. Unlocking the vault enables access to application data off-line or on-line. In contrast, Hybrid Apps can require user credentials that must be checked against Unwired Server on-line before granting access to Hybrid App content.

The password is initially defined when you configure the property values required to enable an authenticated HTTPS connection. However, you can allow users to change this password. For

information about password definition see *changePassword* in the Developer guide for your application type.

See also

- *Securing Sensitive Data On-Device with Data Vault* on page 128

Provisioning Security Artifacts

Typically, you must provision Unwired Platform security artifacts before applications can be used. The manner by which an artifact is provisioned depends on the artifacts themselves, the device types used, and your deployment environment.

Provisioning methods for different application types is extensively documented in *Mobile Application Provisioning*. For complete details on provisioning techniques, read *Stage 3: Provision*.

See also

- *Modifying Default Synchronization Listener Properties with Production Values* on page 104
- *Establishing Encrypted Application Connections* on page 132
- *Unwired Server and Device Application Communications* on page 4
- *Encrypting Synchronization for Replication Payloads* on page 101

Security Artifacts That Require Provisioning

Certain artifacts must be provisioned to the device before the device can connect to the runtime.

- **SSO certificates –** Certificates that identify the user for SSO-enabled applications for SAP backends.
- **Encryption keys –** Public RSA keys that encrypt communication to the Messaging Server.
- **Security profiles –** Profiles saved to the device and contain sensitive information.
- **User and application login credentials –** User names and password that allow a user to access backend data.
- **Connection properties –** The values for server, host, port, Relay Server farm and other property values that can all be provisioned by Afaria. The device user has can also manually enter these values upon initial connection, and these values are stored in the data vault.

See also

- *Provisioning the Public RSA key from the Messaging Server for MBS Encryption* on page 132

Provisioning the Public RSA key from the Messaging Server for MBS Encryption

If you are not using Afaria, you can install the client application, then connect to the corporate LAN using Wi-Fi or other method of your choosing in order to provision devices with required files. This allows you to seed public RSA keys to the device so that over-the-air connections to Unwired Server can be mutually-authenticated and you can minimize the possibility of a rogue server intercepting your initial synchronization and providing its own RSA public key.

Follow these steps to ensure that the public RSA key required for future secure communication is correctly and reliably installed.

1. Provision the application to the device.
2. Connect to the corporate LAN on which the Unwired Server cluster is installed.
3. Use a device connection that connects directly to Unwired Server. Alternatively, you can also connect using the Relay Server settings, but only if Relay Server is accessible from the corporate LAN; typically it is deployed on the DMZ
   Unwired Server seeds the client with the public key. The client uses this public key for all subsequent connections.
4. Provide the user with instructions to reconfigure the connection properties on the device to use Relay Server from the Internet for subsequent connections.

See also

- *Security Artifacts That Require Provisioning* on page 131

Establishing Encrypted Application Connections

Synchronization and messaging connection are encrypted by default. However, for replication connections that use E2EE, the client must be configured correctly to establish connections to the correct HTTP or HTTPS port.

See also

- *Provisioning Security Artifacts* on page 131

Connecting to the TLS Relay Server Port with Client APIS

(Applies only to Windows Mobile and Android devices with replication packages). With the Relay Server environment configured, developers can set appliction client properties to connect to it via the correct port using the TLS protocol.

Note: If Relay Server uses HTTPS and certificates, clients other than those using replication-based synchronization may not be able to connect: messaging applications support only HTTP, and Hybrid Web Container applications for iOS support HTTPS, but not certificates.

1. Ensure the application code has been modified to use the correct TLS protocol, port, and stream parameters, for example:
   - Port – 443
   - Protocol – TLS
   - Stream parameter – `"url_suffix=/ias_relay_server/client/rs_client.dll/[SUP_FARM_ID];tls_type=RSA;trusted_certificates=rsa_root.crt;identity=id_client.pem;identity_password=pwd;"`

   Note: The `identity=id_client.pem;identity_password=pwd` segments of the stream parameter are only required if you use a Relay Server HTTPS port (requires client certificate mutual authentication). This configuration allows the Relay Server to block denial-of-service attacks at the periphery of you network, should you require that degree of security.

   These certificates are personal certificate for the specific user. Typically this file type is not included as part of the application, but separately-installed by the user. In this case, ensure the application prompts the user for the filename and password of that certificate and save it to this parameter.
2. Make the `rsa_root.crt`, and `id_client.pem` (if it is not a personal file the user defines) available for the application on the device. They can be included in the application or deployed separately.

Connecting to the SSL Relay Server Port

Android and Windows Mobile replication clients should connect to Unwired Server through a Relay Server on the DMZ.

Prerequisites

When using E2EE over SSL, certificates need to be created and distributed to both Unwired Server and clients.

Task

Only Windows Mobile can enable E2EE over SSL because these clients use an UltraLite client database. To enable E2EE:

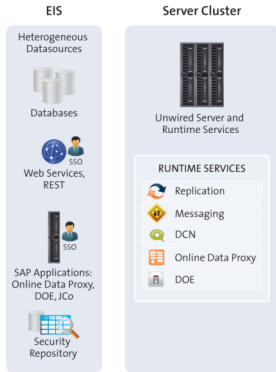
1. Configure application connection code properties to connect to Relay Server with the correct combination of properties.
2. Generate and deploy files that reference this configuration.

For details see *Enable End-to-End Encryption (E2EE) Using SSL* in *Developer Guide: Windows and Windows Mobile Object API Applications*.

CHAPTER 7 EIS Security

There are two aspects to securing interactions between Unwired Platform Runtime and EIS systems.

- **Device application EIS connections –** Whenever a connection to the EIS is configured as an endpoint, Sybase recommends that you use an encrypted connection. To encrypt these connection established by an application, configure the security settings in the connection properties or connection templates. For details, see *Security Settings* in *Sybase Control Center for Sybase Unwired Platform*.
- **Notification communications –** Indirect injections from the EIS to the cache database must be secured. These runtime injections are known as data change notifications (DCNs). You must authenticate a DCN event before the transaction occurs. This chapter documents this mechanism.



Securing EIS Operations: DCN and Push

You can secure two EIS operations: data change notifications (DCN) and push (notifications). To completely secure DCNs, you must protect the communication channel as well as authorize the event request being made on Unwired Server.

Depending on your environment, you can secure one or both EIS notification type. Implementations are seperated to allow you configure securtiy so that technical user is authorized for DCN event changes, but not for Push, and another technical user is authorized for Push events, but not DCN.

- **DCN –** (Not applicable to Online Data Proxy) only *Enabling Authorization of DCNs* is a required activity for securing this notification. By default, DCN shares the same certificate and HTTPS listener is Unwired Server and this secure stream is already enabled. You only

need to change this default configuration if your environment needs to implement a unique security profile to meet your organization's stringency requirements. If you need to understand how the DCN feature is implemented and monitored across Unwired Platform, review the contents listed in *Implementation of Data Change Notification*. These topics help you find DCN related information across the documentation set.

- **Push –** operations allow the EIS to send a notification to synchronize data directly to a specific device. It is entirely separate from MBO operations that can also use notification to trigger synchronization. Push events are secured the same way as DCN. Tasks that describe a DCN activity can be used to secure Push notification events.

See also

- *Related DCN Developer and Administrator Tasks* on page 147

Securing DCN Communications

(Recommended for DCN environments only) By default, the installer automatically configures a single security profile for DCN communications with Unwired Server and Unwired Server administration communications with Sybase Control Center. For most deployments, this default setup is sufficient; however, you can create a new security profile with new SSL certificates if you choose.

Because DCN requests are handled by the WebContainer, you must configure your HTTPS listener accordingly. Considerations to bear in mind include:

- If you choose a *_mutual version of a profile, you must provide your own production-ready certificate. Then you must further create a security configuration in addition to this profile, that handles authentication requests with the CertificateValidationLoginModule. This login module inspects the client certificate to ensure it is signed by a trusted CA, has not expired, and optionally has not been revoked via OCSP or CRL checks. If the certificate is valid, Unwired Server extracts the certificate subject, and that becomes the authenticated principal name for the user. The user must also be in the corresponding DCN User logical role. See *Enabling Authorization of DCNs* and *Certificate Validation Properties*.
- If you choose a non-mutual version of the profile, then know that the client sends BASIC (username/password) credentials. Create a security configuration that uses any module that can authenticate users with that sort of credentials, as well as retrieve physical role membership from the backend security store.

Note: If you are connecting with Online Data Proxy or DOE-C, then each type of connection requires it's own security profile, and the DCN listener profile should not be used in this case.

For details about configuring a new security profile for a custom HTTPS listener for DCN, see *Configuring SSL Properties* in *Sybase Control Center for Sybase Unwired Platform*

See also

- *Enabling Authorization of EIS Operations* on page 137

Enabling Authorization of EIS Operations

All EIS operations must be authorized before the event is processed. Unwired Server authorizes these insertion events using either SUP DCN User or SUP Push User role and a security configuration that names the security provider that performs the authorization function.

The security configuration you create and assign to either DCN package or Push domains can include providers of different types, depending on what you map your role to and the security strength you require.

1. Create an authorization provider for the selected security configuration.

   To choose which type of provider you require, review the types documented in *SUP Roles to Support EIS Operations: SUP DCN User and SUP Push User*. Then follow the instructions in the corresponding *Setting Up Authorization* topic.
2. Ensure that you stack the provider in the order required by your environment.

   This is especially required in SSO-enabled environments. See *Stacking Providers to Authenticate using SSO Before Authorizing*.

See also

- *Securing DCN Communications* on page 136

SUP Roles to Support EIS Operations: SUP DCN User and SUP Push User

SUP DCN User SUP Push User roles are the mechanisms by which illicit EIS DCN or push notification operations are prevented. Like other built-in platform roles, SUP DCN User and SUP Push User are logical roles that are available to all new security configurations.

Before any DCN event is submitted, the person or group mapped to this role must be authorized (after first being authenticated) by a security provider you define as part of a named security configuration. Submitted DCN events that require authorization include:

- Cache updates
- Operation performance

The SUP Push user role is mandatory; with this role the EIS cannot deliver push notifications to Unwired Server for a registered application connection. Before any push event is submitted by the EIS, the authenticated user performing the push must be authorized by being in the SUP Push User logical role. Push events that require authorization include:

- Triggering a Hybrid App package

You can choose different physical role mapping targets to authorize, or authenticate and authorize EIS events using the logical roles. Depending the authorization method used, the implementation varies:

- **Certificate authorization –** Sybase recommends that you use CertificateValidationLoginModule for maximum security.

CertificateValidationLoginModule validates the user certificate passed during mutual certificate authentication. Unlike other methods, it confers no physical roles; therefore, the platform administrator must create a logical role mapping. Typically, the user has a certificate that includes a Subject distinguished name containing a common name (cn=TechnicalUser), so creates a logical role mapping between the logical role and `user:TechnicalUser` in the CN. To implement certificate authorization, see *Setting Up Authorization with Certificate Validation* in *Security*.

Note: While explicitly mapping a certificate user name for SUP Push User role in \Sybase Control Center, ensure there is a space after every comma. Example: `user: CN:PushTest, OU=SSL Server, O=SAP-AG, C=DE"`. Further more, if you are using push notification with strong mutual authentication, you can only use the "Admin" security configuration. Ensure you add a CertificateValidationLoginModule to the Admin security configuration and and us it as the default security configuration in the push-enabled domain. If any other security configuration is used, a `user not in Required role` error is generated in the client log.

- **Technical user authorization –** if the role cannot be mapped to a real user in the security repository of the configured security provider used by the security configuration, you may need to create a new technical user or use an existing technical user for EIS operation role mappings. In this case, no authentication is required as the user is not a real user in the traditional sense. To implement technical user authorization, Sybase recommends that you create a security configuration that includes an LDAP provider. To implement technical user authentication, see *Setting Up Authorization with a Technical User Role Stored in a Repository* in *Security*.
- **Real user authorization –** (Applies only to DCN) if the role must be mapped to a real user, you can authenticate and authorize the user mapped to the SUP DCN User role. You can also use PreconfiguredUserLogin module to perform HTTP Basic authentication, where the module extracts the user information from the request parameter in a URL. To implement real user authentication, see *Setting Up Authorization with PreConfiguredUserLogin Values* in *Security*

Once you have multiple providers configured, especially when implementing authorization with single sign-on, you can stack them so they are processed in correct order. See *Stacking Providers to Authenticate using SSO Before Authorizing* in *Security*.

See also

- *Setting Up Authorization with a Technical User Role Stored in a Repository* on page 139
- *Setting Up Authorization with Certificate Validation* on page 141
- *Setting Up Authorization with PreConfiguredUserLogin Values* on page 144
- *Stacking Providers for DCN SSO Authentication* on page 146

DCN Authorization Considerations for Hybrid App Packages

Hybrid App packages have a unique implementation that bears noting before you begin setting up Sybase Unwired Platform DCN user authorization.

If the workflow DCN sender is authenticated against the default "admin" security configuration, the sender is automatically authorized to push data to all users regardless of their individual security configuration. If not, the sender can push only to users within the same security configuration. For more information, see *Workflow Development for Data Change Notification*.

Setting Up Authorization with a Technical User Role Stored in a Repository

Before any EIS event is submitted, the technical user mapped to either the SUP DCN User or SUP Push User role must be authorized by a security provider you define as part of a named security configuration.

1. *Adding a Physical Technical User Role to Your Security Repository*

   Often the DCN or Push user is not a real user in your security repository, but rather an artificial "technical" user whose credentials are simply a shared secret between the EIS developer writing the DCN client or Push code, and the platform administrator.
2. *Updating an Existing Security Configuration for EIS Event Authorization*

   Platform administrators create different security configurations to authenticate packages that are deployed to various domains. You can revise these existing security configurations to also authenticate EIS events.
3. *Adding a New Provider to Authorize EIS Events*

   Add an authorization provider used to authorize events for packages that use DCN or Push.
4. *Mapping a Sybase Unwired Platform Logical Role to a Technical User in a Repository*

   Mapping the logical role binds it to another technical user's physical ID.

See also

- *SUP Roles to Support EIS Operations: SUP DCN User and SUP Push User* on page 137
- *Setting Up Authorization with Certificate Validation* on page 141
- *Setting Up Authorization with PreConfiguredUserLogin Values* on page 144
- *Stacking Providers for DCN SSO Authentication* on page 146

Adding a Physical Technical User Role to Your Security Repository

Often the DCN or Push user is not a real user in your security repository, but rather an artificial "technical" user whose credentials are simply a shared secret between the EIS developer writing the DCN client or Push code, and the platform administrator.

If a suitable technical user does not exist, Sybase recommends that you add one to the repository or user registry. Different providers have different methods for achieving this goal.

Talk to the administrator of that repository and agree on the user to use for the logical role mapping.

- **For LDAP –** There are different ways to add users, including technical users. However, the easiest is to create an LDAP Data Interchange Format (LDIF) file. The file contains a set of users to be added into the directory. The file is used by common LDAP utilities, such as ldapModify.

Otherwise, to map the logical user role to a physical one, manually enter a user ID using a predefined syntax. To avoid errors, ensure that you closely follow the prescribed syntax rules. For details, see *Mapping a Logical Role to a Technical User in a Repository* later in this sequence.

Updating an Existing Security Configuration for EIS Event Authorization

Platform administrators create different security configurations to authenticate packages that are deployed to various domains. You can revise these existing security configurations to also authenticate EIS events.

When choosing a security configuration to modify, determine which existing security configurations are assigned to them. Add or edit login modules or authorizers that a technical user can be authenticated with before being conferred the logical role used by Unwired Platform.

1. In the left navigation pane of Sybase Control Center, select **Security**.
2. Select a security configuration
3. Click **Edit**.

Adding a New Provider to Authorize EIS Events

Add an authorization provider used to authorize events for packages that use DCN or Push.

1. In the navigation pane of Sybase Control Center, open the security configuration that is assigned to the DNC package or Push domain, then click **New**.
2. Select the provider you want to add.
3. Configure the properties associated with the provider by:
   - Setting values for available properties according to your security requirements.
   - Adding new properties and corresponding values as required.

   For more information about configuring security provider properties, see the individual reference topics for each provider type.
4. Set the Control Flag property to `sufficient`, to ensure this provider is processed and ordered correctly.
5. Click **OK**.

Mapping a Sybase Unwired Platform Logical Role to a Technical User in a Repository

Mapping the logical role binds it to another technical user's physical ID.

1. In the navigation pane of Sybase Control Center, select the security configuration that is assigned to the DCN or Push domain.
2. Locate the logical role you want to map, then choose an appropriate action:
   - If the logical role exactly matches the technical user role name in the security provider repository, select **AUTO**.
   - If the logical role differs from the technical user role name in the security provider repository, select **Map Role**, then select the technical role name in **Available roles** and click **Add**.

     Note: If you are using ActiveDirectory, and are using an e-mail address for the technical user names, the definition appears as *username@myaddress@DomainSecurityConfigName*.
   - To map the role name to a specific user name rather than a role, select **Map Role**, then type the name in **Available roles** as `user:TechnicalUser` and click **Add**.

The logical role now shows the mapping state changes to MAPPED.

Setting Up Authorization with Certificate Validation

(Recommended) Before any event is submitted to the Sybase Unwired Platform runtime, the subject of the certificate's CN that is mapped to the logical role must be authorized, and with Certificate Validation, authenticated. Sybase recommends using a CertificateValidationLoginModule for mutual authentication. This solution offers maximum DCN or Push security.

1. *Updating an Existing Security Configuration for EIS Event Authorization*

   Platform administrators create different security configurations to authenticate packages that are deployed to various domains. You can revise these existing security configurations to also authenticate EIS events.
2. *Adding a certificateValidationLoginModule for DCN Mutual Authentication*

   For DCN events, you can add a certificateValidationLoginModule to perform mutual authentication, and stack it with other modules that perform the authentication of device users.
3. *Mapping Logical Role to the Subject for the Certificate CN*

   The certificate used for mutual authentication includes a common name (CN) that is extracted and compared to the physical role mapping you create using this CN.

See also

- *SUP Roles to Support EIS Operations: SUP DCN User and SUP Push User* on page 137

- *Setting Up Authorization with a Technical User Role Stored in a Repository* on page 139
- *Setting Up Authorization with PreConfiguredUserLogin Values* on page 144
- *Stacking Providers for DCN SSO Authentication* on page 146

Updating an Existing Security Configuration for EIS Event Authorization

Platform administrators create different security configurations to authenticate packages that are deployed to various domains. You can revise these existing security configurations to also authenticate EIS events.

When choosing a security configuration to modify, determine which existing security configurations are assigned to them. Add or edit login modules or authorizers that a technical user can be authenticated with before being conferred the logical role used by Unwired Platform.

1. In the left navigation pane of Sybase Control Center, select **Security**.
2. Select a security configuration
3. Click **Edit**.

Adding a certificateValidationLoginModule for DCN Mutual Authentication

For DCN events, you can add a certificateValidationLoginModule to perform mutual authentication, and stack it with other modules that perform the authentication of device users.

Prerequisites

- Use the same installer provided HTTPS certificates used for Unwired Server and Sybase Control Center mutual authentication (as part of the SSL security profile). If you use new certificates exclusively for DCN, import all required files to the correct trust and key stores.
- Use certificateValidationLoginModue for mutual authentication by creating a new HTTPS listener exclusively for DCN communications and choosing one of the three mutual authentication types (domestic_mutual, strong_mutual , or intl_mutual).

See *Securing DCN Communications*.

Task

1. In the navigation pane of Sybase Control Center, open the security configuration that is assigned to the DCN package domain, then click **New**.
2. Select **certificateValidationLoginModule**.
3. Configure the properties associated with the provider. See *Certificate Validation Properties*.
4. Set the Control Flag property to sufficient, to ensure this provider is processed and ordered correctly.

5. Share configured property values with the EIS developers who are writing the DCN sending logic.

See also

- *Certificate Validation Properties* on page 166

Mapping Logical Role to the Subject for the Certificate CN

The certificate used for mutual authentication includes a common name (CN) that is extracted and compared to the physical role mapping you create using this CN.

CertificateValidationLoginModule validates the user certificate passed during mutual certificate authentication. Unlike other methods, it confers no physical roles. Therefore, the platform administrator must create a logical role mapping. A CN of a certificate typically looks like:

```
CN=TechnicalUser, OU=sybase, O=sap
```

Optionally, you can use the entire subject as the user name, meaning the whole CN is included, for example, `user:CN=TechicalUser, OU=sybase, O=sap`.

When using the certificate, ensure the **Validated certificate is identity property** of CertificateValidationLoginModule is set to true. Also ensure the user maps the entire subject name to the logical role, instead of the CN value.

If you are supporting multiple domains, the mapped user name must also include the named security configuration for either the package the DCN is targeted for or the Admin security configuration for of a Push domain, and appended as a *@DomainSecurityConfigName* suffix.

For example, uppose you have two packages (PKG_A, PKG_B) deployed to two domains (Domain_A, Domain_B) respectively. PKG_A in Domain_A has been assigned to the DCN security configuration, and PKG_B in Domain_B has been assigned to the "DCN2SecurityConfig" security configuration.

- A DCN event for PKG_A is authorized with *TechnicalUser@DCNSecurity*.
- A DCN event for PKG_B is authorized with *TechnicalUser@DCN2SecurityConfig*.

1. In the navigation pane, select the security configuration you have created and assigned to the DCN package domain.
2. Click the **SUP DCN User** role, and select **Map Role**.
3. In Role name, enter the physical role as `user:TechnicalUser`, then click **+**.

   Repeat this action for each unique common name of different DCN users.
4. Click **Add** to move the role to the Mapped Roles column.
5. Click **OK**.

The SUP DCN User role now shows the mapping state changes to MAPPED.

Setting Up Authorization with PreConfiguredUserLogin Values

Before any EIS event is submitted to Unwired Server, the person or group mapped to the SUP DCN User role must be authorized and authenticated. You can use the PreConfiguredUserLogin module with HTTP Basic authentication for this purpose.

1. *Updating an Existing Security Configuration for EIS Event Authorization*

   Platform administrators create different security configurations to authenticate packages that are deployed to various domains. You can revise these existing security configurations to also authenticate EIS events.
2. *Adding a PreconfiguredUserLoginModule for HTTP Basic Authentication*

   You can add the PreconfiguredUserLoginModule to perform HTTP Basic authentication for a puch or DCN events, and stack it with other modules that perform the authentication of device users.
3. *Mapping DCN or Push Roles to a User Name Defined In PreconfiguredUserLoginModule*

   Map either the SUP DCN User or SUP Push User logical role to the non-standard value you supplied.

See also

- *SUP Roles to Support EIS Operations: SUP DCN User and SUP Push User* on page 137
- *Setting Up Authorization with a Technical User Role Stored in a Repository* on page 139
- *Setting Up Authorization with Certificate Validation* on page 141
- *Stacking Providers for DCN SSO Authentication* on page 146

Updating an Existing Security Configuration for EIS Event Authorization

Platform administrators create different security configurations to authenticate packages that are deployed to various domains. You can revise these existing security configurations to also authenticate EIS events.

When choosing a security configuration to modify, determine which existing security configurations are assigned to them. Add or edit login modules or authorizers that a technical user can be authenticated with before being conferred the logical role used by Unwired Platform.

1. In the left navigation pane of Sybase Control Center, select **Security**.
2. Select a security configuration
3. Click **Edit**.

Adding a PreconfiguredUserLoginModule for HTTP Basic Authentication

You can add the PreconfiguredUserLoginModule to perform HTTP Basic authentication for a puch or DCN events, and stack it with other modules that perform the authentication of device users.

The PreconfiguredUserLoginModule is typically used to give the Platform administrators access to Sybase Control Center, so that this individual can log in securely and configure the runtime immediately upon installation. Once logged in, administrators are expected to immediately replace this login module in the "admin" security configuration on the "default" domain. Once PreconfiguredUserLoginModule is replaced, you can use this login module Push or DCN events as an alternative to HTTPS Basic authentication. In this scenario, this login module extracts the user information from a request parameter in a URL written with this format:

`http://<hostname>:<port>/dcn/HttpAuthDCNServlet`.

See *Basic HTTP Authentication* in the *Mobile Data Models: Using Mobile Business Objects* guide.

1. In the navigation pane of Sybase Control Center, open the security configuration for either Push or DCN events, then click **New**.
2. Select **PreConfiguredUserLoginModule**.
3. Configure the properties associated with the provider by configuring values:
   - **User name –** configure the value appropriately fore these security provider types:
     - If you are using HTTPS Basic, ensure that the user name specified in the HTTPS Basic authentication response needs is formatted as: `user@security_configuration`. This format ensures that authentication/ authorization happens in the correct security context.
     - In some ActiveDirectory configurations, the username may have the form user@activeDirectoryDomain. If, for example, a DCN user is fred@acme.com, and the security configuration is named "DCN", then the username specified for DCN request must be "fred@acme.com@DCN".
   - **Password –** set a password that corresponds to the user name entered.
   - **Role –** enter `SUP DCN User` or `SUP Push User`.
4. Set the Control Flag property to `sufficient`, to ensure this provider is processed and ordered correctly.
5. Share configured property values with the EIS developers who are writing the DCN sending logic or the Push notification.

See also

- *Assigning Providers to a Security Configuration* on page 53
- *Stacking Providers and Combining Authentication Results* on page 97
- *HTTP Authentication Security Provider* on page 62

- *HTTP Basic Authentication Properties* on page 169
- *Preconfigured User Authentication Properties* on page 177

Mapping DCN or Push Roles to a User Name Defined In PreconfiguredUserLoginModule

Map either the SUP DCN User or SUP Push User logical role to the non-standard value you supplied.

If you used SUP DCN User or SUP Push User role and entered the Role property name so the value exactly matches, the mapping state is **AUTO**.

1. In the navigation pane of Sybase Control Center, open the security configuration that is assigned to the DCN package domain, then click **New**.
2. Click the entry for either role and:
   a) Select **Map Role**.
   b) In the Available roles column, choose the value you configured for the Role property of the provider.
   c) Click **Add**.

The mapping state now changes to MAPPED.

Stacking Providers for DCN SSO Authentication

(Applies only to DCN events) Stack DCN providers with SSO providers to authenticate the SUP DCN User logical role with the SSO mechanisms. The users must be authenicated before they can be authorized.

The providers you use for SSO can vary.

1. For HttpAuthenticationLoginModule SSOimplementations, the CertificateAuthenticationLoginModule must be first in the list with the controlFlag set to "sufficient". If authentication succeeds, no other modules are used unless their controlFlags are set to "required".
2. For CertificateAuthenticationLoginModule implementations, , stack the chose DCN provider after the CertificateAuthenticationLoginModule and:
   a) Set the CertificateAuthenticationLoginModule's controlFlag to "sufficient", and order it first in the stack. This sequence allows normal device users to authenticate quickly.
   b) Choose any other user name and password-based login module to stack with its controlFlag set to either "optional" or "sufficient".

See also

- *SUP Roles to Support EIS Operations: SUP DCN User and SUP Push User* on page 137
- *Setting Up Authorization with a Technical User Role Stored in a Repository* on page 139

- *Setting Up Authorization with Certificate Validation* on page 141
- *Setting Up Authorization with PreConfiguredUserLogin Values* on page 144

Related DCN Developer and Administrator Tasks

DCN is a feature that is performed in different runtime and development components of Unwired Platform. Review the tasks that must be performed by different roles to effectively implement DCN and Hybrid App DCN in an end-to-end environment.

See also

- *Securing EIS Operations: DCN and Push* on page 135

MBO Development for Data Change Notification

Sybase Unwired WorkSpace - Mobile Business Object Development contains details about configuring MBOs to enable DCN to refresh cached MBO data.

While an MBO belongs to a single cache group, MBOs in the same project are not necessarily in the same cache group. The cache group policy determines the data refresh behavior of all MBOs within the group. DCN can be used as the sole mechanism of refreshing cached data in Unwired Server by specifying the DCN cache refresh policy. See *Best Practices for Loading Data from the EIS to the CDB* in *Mobile Data Models: Using Mobile Business Objects*.

See also

- *Hybrid App Development for Data Change Notification* on page 147
- *Management and Monitoring of Data Change Notifications* on page 148

Hybrid App Development for Data Change Notification

Developer Guide: Hybrid Apps contains details about configuring Hybrid Apps to enable DCN to refresh cached Hybrid App data.

Goal	Topic
Implement DCNs for Hybrid Apps.	*Extending Data Change Notification to Hybrid App Clients*
Create DCN authorization requests using PreconfiguredUserLogin values.	*Non HTTP Authentication Hybrid App DCN Request*
Review the structure of a DCN responses.	*Hybrid App DCN Request Response*

See also

- *MBO Development for Data Change Notification* on page 147
- *Management and Monitoring of Data Change Notifications* on page 148

Management and Monitoring of Data Change Notifications

Various Sybase product documentation guides provide information about managing DCNs and monitoring DCN statistics and performance.

Goal	Topic
Use DCN with other cache management features.	*Cache Data Management* in the *Mobile Application Life Cycle*
Configure synchronization groups to notify device users when a DCN event has occurred. The notification tells the user to synchronize and receive those updates.	In *Sybase Control Center for Sybase Unwired Platform*, see *Configuring Synchronization Groups* in *Sybase Control Center for Sybase Unwired Platform*.
Monitor DCN activity.	In *Sybase Control Center for Sybase Unwired Platform*, see *Checking System Statistics*, *Related Data Change Notification Information*.

See also

- *MBO Development for Data Change Notification* on page 147
- *Hybrid App Development for Data Change Notification* on page 147

CHAPTER 8 Security Monitoring and Issue Detection

Identify, analyze, and resolve current and potential Unwired Platform security problems.

Tools and Diagnostic Methodologies

Sybase recommends administrators use of multiple diagnostic sources. Monitoring data can effectively supplement obscure and abbreviated debug/syslog output.

Platform Security Monitoring

Active security monitoring is about catching small problems before they turn into serious issues. It's also about taking proactive steps to protect your Unwired Platform against unnecessary risks. Use Sybase Control Center to help you proactively maintain the defense mechanisms you diligently implemented for your mobile components, data, and resources.

Current, historical, and performance-based tabs allow you to diagnose security health and issues for user support, troubleshooting, and security performance tracking.

Reviewing System Monitoring Data

Review data for monitored activities in Sybase Control Center. The monitoring data is retrieved according to the specified time range. Key Performance Indicators (KPIs) are also calculated for the specified time range.

1. Open and log in to Sybase Control Center.
2. In the left navigation pane, select **Monitoring**.
3. In the right administration pane, select one of the following tabs according to the type of monitoring data you want to view:
   - **Security Log**
   - **Replication**
   - **Messaging**
   - **Queue**
   - **Data Change Notifications**
   - **Device Notifications**
   - **Package Statistics**
   - **User Statistics**
   - **Cache Statistics**

Security Log Statistics

The security log reflects the authentication history of users either across the cluster, or filtered according to domain, during a specified time period. These statistics allow you to diagnose and troubleshoot connection or authentication problems on a per-user basis. Security log monitoring is always enabled.

User security data falls into these categories:

Category	Description
User	The user name
Security Configuration	The security configuration to which the device user belongs
Time	The time at which the authentication request took place
Result	The outcome of the authentication request: success or failure
Application Connection ID	The application connection ID associated with the user
Package	The package the user was attempting to access
Domain	The domain the user was attempting to access

Common Analysis Scenarios

Use these scenarios to walk you through typicaly security issues that may require a particular assessment path.

Access Denied Analysis

If a user reports an `access is denied` error, the administrator can check the security log, and from there, validate the package's security configuration.

Checking the Security Log

Validate `access is denied` messages by checking the security log.

1. In Sybase Control Center, click the Monitoring node.
2. Click **Security Log**.
3. Set the **Start Date**, **Start Time** and **Finish Date**, **Finish Time** to restrict the data to the specified time frame.
4. Click the **Result** column to sort rows by result type.
5. Locate any authentication failures or access denied events that are logged for the user who reported the error.

6. If you find any errors in the result column, check package names and security configurations. Then check to see if a similar result is reported for other user names. Also verify if the error persists; a heavily loaded service could cause a transient error. Transient errors are generally resolved retrying the connection.

Next

If there are no errors, investigate the security setup for the pair.

Validating Security Setup

If users are reporting `access is denied` errors where access should be allowed, validate your security setup. A security configuration is defined at the cluster level by the platform administrator, then assigned at domain and package levels by either administrator type, so it may take some analysis to determine where the problem is occurring.

Use the security log to evaluate the properties of the assigned security configuration.

1. In Sybase Control Center, expand the navigation tree in the Unwired Platform administration perspective until you locate the security configuration that generated the error. It appears either in the **Domains** > **<domain_name>** > **Security** folder or the **Security** folder at the cluster root.
2. Select the configuration you are investigating.
   - If the security configuration is assigned to a domain, validate that the role mapping is correct:
     - If the Unwired Platform user is the exact name of the user in the security repository, then no mapping is required.
     - If the Unwired Platform user differs, even slightly, then logical roles used by the package, and physical roles used in the repository must be manually mapped.
   - Review the existing security policy with the security administrator to ensure that privileges are set correctly.

APPENDIX A Security Reference

Security Provider Configuration Properties

Security providers implement different properties, depending on the type of provider you are configuring.

Platform administrators can configure application security properties in the Sybase Control Center. These properties are then transcribed to an XML file in the *SUP_HOME*\Servers\UnwiredServer\Repository\CSI\ directory. A new section is created for each provider you add.

LDAP Configuration Properties

Use these properties to configure the LDAP provider used to authenticate Sybase Control Center administration logins or to configure the LDAP provider used to authenticate device application logins. If you are configuring an LDAP provider for device application logins in Sybase Control Center, then Unwired Platform administrators use Sybase Control Center these properties are saved to the *SUP_HOME*\Servers\UnwiredServer\Repository\CSI\<security configuration name file.

The Java LDAP provider consists of three provider modules.

- The **LDAPLoginModule** provides authentication services. Through appropriate configuration, you can enable certificate authentication in **LDAPLoginModule**.
- (Optional) **LDAPAuthorizer** or **RoleCheckAuthorizer** provide authorization service in conjunction with LDAPLoginModule. LDAPLoginModule works with either authorizer.

  The **RoleCheckAuthorizer** is part of every security configuration but does not appear in Sybase Control Center.

  Use **LDAPAuthorizer** only when **LDAPLoginModule** is not used to perform authentication, but roles are still required to perform authorization checks against the LDAP data store. If you use **LDAPAuthorizer**, always explicitly configure properties; for it cannot share the configuration options specified for the **LDAPLoginModule**.
- (Optional) **LDAPAttributer** is used to retrieve the list of roles from the LDAP repository. These roles are displayed in the role mapping screen in Sybase Control Center. The LDAP attributer is capable of sharing the configuration properties from the LDAPLoginModules. If no configuration properties are explicitly specified, then the attributer iterates through the configured LDAPLoginModules and retrieves the roles from all the LDAP repositories configured for the different LDAPLoginModules.

Use this table to help you configure properties for one or more of the supported LDAP providers. When configuring modules or general server properties in Sybase Control Center, note that properties and values can vary, depending on which module or server type you configure.

Property	Default Value	Description
ServerType	None	Optional. The type of LDAP server you are connecting to: • `sunone5` -- SunOne 5.x OR iPlanet 5.x • `msad2k` -- Microsoft Active Directory, Windows 2000 • `nsds4` -- Netscape Directory Server 4.x • `openldap` -- OpenLDAP Directory Server 2.x The value you choose establishes default values for these other authentication properties: • RoleFilter • UserRoleMembership • RoleMemberAttributes • AuthenticationFilter • DigestMD5Authentication • UseUserAccountControl
ProviderURL	`ldap://localhost:389`	The URL used to connect to the LDAP server. Without this URL configured, Unwired Server cannot contact your server. Use the default value if the server is: • Located on the same machine as your product that is enabled with the common security infrastructure. • Configured to use the default port (389). Otherwise, use this syntax for setting the value: `ldap://<hostname>:<port>`

Property	Default Value	Description
DefaultSearchBase	None	The LDAP search base that is used if no other search base is specified for authentication, roles, attribution and self registration: **1.** `dc=<domainname>,dc=<tld>` For example, a machine in sybase.com domain would have a search base of dc=sybase,dc=com. **2.** `o=<company name>,c=<country code>` For example, this might be o=Sybase,c=us for a machine within the Sybase organization. **Note:** When you configure this property in the "admin" security configuration used to authenticate the administrator in Sybase Control Center, the property value should not contain any special characters, as listed above, in any of the common names or distinguished names.
SecurityProtocol	None	The protocol to be used when connecting to the LDAP server. The specified value overrides the environment property java.naming.security.protocol. To use an encrypted protocol, use SSL instead of ldaps in the URL.
AuthenticationMethod	Simple	The authentication method to use for all authentication requests into LDAP. Legal values are generally the same as those of the java.naming.security.authentication JNDI property. Choose one of: • simple — For clear-text password authentication. • DIGEST-MD5 — For more secure hashed password authentication. This method requires that the server use plain text password storage and only works with JRE 1.4 or later.

Property	Default Value	Description
AuthenticationFilter	For most LDAP servers: `(&(uid={uid})(objectclass=person))` or For Active Directory e-mail lookups: `(&(userPrincipalName={uid})(objectclass=user))[ActiveDirectory]` For Active Directory Windows user name lookups: `(&(sAMAccountName={uid})(objectclass=user))`	The filter to use when looking up the user. When performing a user name based lookup, this filter is used to determine the LDAP entry that matches the supplied user name. The string "{uid}" in the filter is replaced with the supplied user name. **Note:** When you use this property to authenticate a user in Sybase Control Center: • The property value should not contain any special characters, as listed above, in any of the common names or distinguished names. • Do not use Chinese or Japanese characters in user names or passwords of this property.
AuthenticationScope	onelevel	The authentication search scope. The supported values for this are: • `onelevel` • `subtree` If you do not specify a value or if you specify an invalid value, the default value is used.
AuthenticationSearchBase	None	The search base used to authenticate users. If this property is not configured, the value for DefaultSearchBase is used. **Note:** When you configure this property in the "admin" security configuration used to authenticate the administrator in Sybase Control Center, the property value should not contain any special characters, as listed above, in any of the common names or distinguished names.

Property	Default Value	Description
BindDN	None	The user DN to bind against when building the initial LDAP connection. In many cases, this user may need read permissions on all user records. If you do not set a value, anonymous binding is used. Anonymous binding works on most servers without additional configuration.

BindPassword	None	The password for BindDN, which is used to authenticate any user. BindDN and BindPassword separate the LDAP connection into units.
		
		The AuthenticationMethod property determines the bind method used for this initial connection.
		
		Sybase recommends that you encrypt passwords, and provides a password encryption utility. If you encrypt BindPassword, include `encrypted=true` in the line that sets the option. For example:
		`<options name="BindPassword" encrypted="true" value="1snjikfwregfqr43hu5io..."/>`
		
		If you do not encrypt BindPassword, the option might look like this:
		`<options name="BindPassword" value="s3cr3T"/>`

Property	Default Value	Description
RoleSearchBase	None	The search base used to retrieve lists of roles. If this property is not configured, the value for DefaultSearchBase is used. **Note:** Setting the RoleSearchBase to the root in Active Directory (for example "DC=example,DC=com") results in a PartialResultsException error when validating the configuration or authenticating a user. To confirm, verify that example.com:389 is reachable. The DNS lookup may successfully resolve example.com to an IP address but port 389 may not be open with an Active Directory server listening on that port. In this case, adding an entry to the `systemroot\system32\drivers\etc\hosts` (typically, `C:\WINDOWS\system32\drivers\etc\hosts`) file on the machine where Sybase Unwired Platform is installed resolves any communication error. **Note:** When you configure this property in the "admin" security configuration used to authenticate the administrator in Sybase Control Center, the property value should not contain any special characters, as listed above, in any of the common names or distinguished names.

Property	Default Value	Description
RoleFilter	For SunONE/iPlanet: `(&(objectclass=ldapsubentry) (objectclass=nsroledefinition))` For Netscape Directory Server: `(|(objectclass=groupofnames) (objectclass=groupofuniquenames))` For ActiveDirectory: `(|(objectclass=groupofnames) (objectclass=group))`	The role search filter. This filter should, when combined with the role search base and role scope, return a complete list of roles within the LDAP server. There are several default values, depending on the chosen server type. If the server type is not chosen and this property is not initialized, no roles are available. **Note:** When you use this property to authenticate a user in Sybase Control Center: • The property value should not contain any special characters, as listed above, in any of the common names or distinguished names. • Do not use Chinese or Japanese characters in user names or passwords of this property.
RoleMemberAttributes	For Netscape Directory Server and OpenLDAP Server: member,uniquemember	A comma-separated list of role attributes from which LDAP derives the DNs of users who have this role. These values are cross-referenced with the active user to determine the user's role list. One example of the use of this property is when using LDAP groups as placeholders for roles. This property has a default value only when the Netscape server type is chosen.
RoleNameAttribute	cn	The attribute of the role entry used as the role name in Unwired Platform. This is the role name displayed in the role list or granted to the authenticated user.
RoleScope	onelevel	The role search scope. Supported values include: • `onelevel` • `subtree` If you do not specify a value or if you specify an invalid value, the default value is used.

Property	Default Value	Description
SkipRoleLookup	false	Set this property to true to grant the roles looked up using the attributes specified by the property UserRoleMembershipAttributes without cross-referencing them with the roles looked up using the RoleSearchBase and RoleFilter. LDAP configuration validation succeeds even when an error is encountered when listing all the available roles. The error is logged to the server log during validation but not reported in Sybase Control Center, allowing the configuration to be saved. This has an impact when listing the physical roles for role mapping as well as in Sybase Control Center. To successfully authenticate the user, set the SkipRoleLookup property to true.
UserRoleMembershipAttributes	For iPlanet/SunONE: nsRoleDN For Active Directory: memberOf For all others: none	Defines a user attribute that contains the DNs of all of the roles a user is a member of. These comma-delimited values are cross-referenced with the roles retrieved in the role search base and search filter to generate a list of user's roles. If SkipRoleSearch property is set to true, these comma-delimited values are not cross-referenced with the roles retrieved in the role search base and role search filter. See *Skipping LDAP Role Lookups (SkipRoleLookup)*. **Note:** If you use nested groups with Active Directory, you must set this property to tokenGroups. See *LDAP Nested Groups and Roles in LDAP*.
UserFreeformRoleMembershipAttributes	None	The freeform role membership attribute list. Users who have attributes in this comma-delimited list are automatically granted access to roles whose names are equal to the attribute value. For example, if the value of this property is department and user's LDAP record has the following values for the department attribute, { sales, consulting }, then the user will be granted roles whose names are sales and consulting.

Property	Default Value	Description
Referral	ignore	The behavior when a referral is encountered. Valid values are dictated by LdapContext, for example, follow, ignore, throw.
DigestMD5Authentication-Format	DN For OpenLDAP: User name	The DIGEST-MD5 bind authentication identity format.
UseUserAccountControlAttribute	For Active Directory: true	When this property is set to true, the UserAccountControl attribute is used to detect if a user account is disabled, if the account has expired, if the password associated with the account has expired, and so on. Active Directory uses this attribute to store this information.
controlFlag	optional	When you configure multiple authentication providers, use controlFlag for each provider to control how the authentication providers are used in the login sequence. controlFlag is a generic login module option rather than an LDAP configuration property. For more information, see *controlFlag Attribute Values*.

See also

- *LDAP Security Provider* on page 56
- *Adding a Production-Grade Provider* on page 33
- *Stacking Providers and Combining Authentication Results* on page 97

NTProxy Configuration Properties

(Not applicable to Online Data Proxy) Configure these properties to allow the operating system's security mechanisms to validate user credentials using NTProxy (Windows Native OS). Access these properties from the Authentication tab of the Security node in Sybase Control Center.

Table 6. Authentication properties

Properties	Default Value	Description
Extract Domain From Username	true	If set to true, the user name can contain the domain in the form of *<username>@<domain>*. If set to false, the default domain (described below) is always used, and the supplied user name is sent to through SSPI untouched.
Default Domain	The domain for the host computer of the Java Virtual Machine.	Specifies the default host name, if not overridden by the a specific user name domain.
Default Authentication Server	The authentication server for the host computer of the Java Virtual Machine.	The default authentication server from which group memberships are extracted. This can be automatically determined from the local machine environment, but this property to bypass the detection step.
useFirstPass	false	If set to true, the login module attempts to retrieve only the user name and password from the shared context. It never calls the callback handler.
tryFirstPass	false	If set to true, the login module first attempts to retrieve the user name and password from the shared context before attempting the callback handler.
clearPass	false	If set to true, the login module clears the user name and password in the shared context when calling either commit or abort.
storePass	false	If set to true, the login module stores the user name and password in the shared context after successfully authenticating.

NoSecurity Configuration Properties

A NoSecurity provider offers pass-through security for Unwired Server, and should be typically be reserved for devlopment or testing. Sybase strongly encourages you to avoid

using this provider in production environments — either for administration or device user authentication.

- The NoSecLoginModule class provides open authentication services
- The NoSecAuthorizer class provides authorization services
- The NoSecAttributer provides attribution services

You need to configure only the authentication properties for the NoSecurity provider.

Table 7. Authentication Properties

Property	Default Value	Description
useUsernameAsIdentity	true	If this option is set to true, the user name supplied in the callback is set as the name of the principal added to the subject.
identity	nosec_identity	The value of this configuration option is used as the identity of the user if either of these conditions is met: • No credentials were supplied. • The useUsernameAsIdentity option is set to false.
useFirstPass	false	If set to true, the login module attempts to retrieve only the user name and password from the shared context. It never calls the callback handler.
tryFirstPass	false	If set to true, the login module first attempts to retrieve the user name and password from the shared context before attempting the callback handler.
clearPass	false	If set to true, the login module clears the user name and password in the shared context when calling either commit or abort.
storePass	false	If set to true, the login module stores the user name and password in the shared context after successfully authenticating.

Note: When you create a new security configuration, Sybase Unwired Platform sets the NoSecurity provider by default. Sybase recommends that after you add, configure, and

validate your providers, you remove the NoSecurity provider. For more information, see *Creating a Security Configuration*.

Certificate Authentication Properties

Add and configure authentication provider properties for CertificateAuthenticationLoginModule, or accept the default settings.

Note: This provider cannot be used for administrative security (in the "admin" security configuration).

Table 8. CertificateAuthenticationLoginModule properties

Property	Description
Implementation class	The fully qualified class that implements the login module. `com.sybase.security.core.CertificateAuthenticationLoginModule` is the default class.
Provider type	`LoginModule` is the only supported value.
Control flag	Determines how success or failure of this module affects the overall authentication decision. `optional` is the default value.
Clear password	(Optional) If true, the login module clears the user name and password from the shared context. The default is false.
Store password	(Optional) If true, the login module stores the user name and password in the shared context. The default is false.
Try first password	(Optional) If true, the login module attempts to retrieve user name and password information from the shared context, before using the callback handler. The default is false.
Use first password	(Optional) If true, the login module attempts to retrieve the user name and password only from the shared context. The default is false.
Enable revocation checking	(Optional) Enables online certificate status protocol (OCSP) certificate checking for user authentication. If you enable this option, you muse enable OCSP in Unwired Server. This provider uses the values defined as part of the SSL security profile. Revoked certificates result in authentication failure when both of these conditions are met: • revocation checking is enabled • OCSP properties are configured correctly

Property	Description
Regex for username certificate match	(Optional) By default, this value matches that of the certificates common name (CN) property used to identify the user. If a mobile application user supplies a user name that does not match this value, authentication fails.
Trusted certificate store	(Optional) The file containing the trusted CA certificates (import the issuer certificate into this certificate store). Use this property and `Store Password` property to keep the module out of the system trust store. The default Unwired Server system trust store is *`SUP_HOME`*`\Servers\UnwiredServer\Repository\Securitytruststore\truststore.jks`. If you do not specify a store location:: • Unwired Server checks to see if a store used by the JVM (that is, the one defined by the `javax.net.ssl.trustStoreType` system property. • If the system property is not defined, then this value is used: `${java.home}/lib/security/jssecacerts` • If that location also doesn't exist, then this value is used: `${java.home}/lib/security/cacerts` **Note:** This property is required only if Validate certificate path is set to true.
Trusted certificate store password	(Optional) The password required to access the trusted certificate store. For example, import the issuer of the certificate you are trying to authenticate into the shared JDK cacerts file and specify the password using this property. **Note:** This property is required only if Validate certificate path is set to true. However, you do not need to configure this value if the default is used. The default value is the value of the `javax.net.ssl.trustStorePassword` property.

Property	Description
Trusted certificate store provider	(Optional) The keystore provider. For example, "SunJCE." **Note:** This property is required only if Validate certificate path is set to true. However, you do not need to configure this value if the default is used. The default value is the value of the `javax.net.ssl.trustStoreProvider` property. If it is not defined, then the most preferred provider from the list of registered providers that supports the specified certificate store type is used.
Trusted certificate store type	(Optional) The type of certificate store. For example, "JKS." **Note:** This property is required only if Validate certificate path is set to true. However, you do not need to configure this value if the default is used. The default value is the value of the `javax.net.ssl.trustStore` property. If this value is not defined, then default value is the keystore type as specified in the Java security properties file, or the string "jks" (Java keystore) if no such property exists.
Validate certificate path	If true (the default), performs certificate chain validation of the certificate being authenticated, starting with the certificate being validated. Verifies that the issuer of that certificate is valid and is issued by a trusted certificate authority (CA), if not, it looks up the issuer of that certificate in turn and verifies it is valid and is issued by a trusted CA. In other words, it builds up the path to a CA that is in the trusted certificate store. If the trusted store does not contain any of the issuers in the certificate chain, then path validation fails. For information about adding a certificate to the truststore, see *Using Keytool to Generate Self-Signed Certificates and Keys* in *Security*.

See also

- *Certificate Security Provider* on page 61
- *SAP SSO Token Security Provider* on page 61
- *HTTP Authentication Security Provider* on page 62
- *Using Keytool to Generate Self-Signed Certificates and Keys* on page 43

Certificate Validation Properties

Add and configure provider properties for CertificateValidationLoginModule, or accept the default settings. CertificateValidationLoginModule can be used in conjunction with other

login modules that support certificate authentication (for example, LDAPLoginModule) by configuring CertificateValidationLoginModule before the login modules that support certificate authentication.

You can only use this provider to validate client certificates when an HTTPS listeners is configured to use mutual authentication.

Table 9. CertificateValidationLoginModule properties

Property	Description
Implementation class	The fully qualified class that implements the login module. `com.sybase.security.core.CertificateValidationLoginModule` is the default class.

crl.[index].uri	Specifies the universal resource identifier for the certificate revocation list (CRL). Multiple CRLs can be configured using different values for the index. The CRLs are processed in index order. For example:
	`crl.1.uri=http://crl.verisign.com/ThawtePersonalFreemailIssuingCA.crl`
	`crl.2.uri=http://crl-server/`

Provider type	`LoginModule` is the only supported value.
Validated certificate is identity	(Optional) Determines if the certificate should be set the authenticated subject as the user ID. If the CertificateValidationLoginModule is used in conjunction with other login modules that establish user identity based on the validated certificate, set this value to `false`. If you are implementing this provider with a DCN security configuration, and it's also not used with SSO, then set this property to `true`.`False` is the default value.
Enable revocation checking	(Optional) Enables online certificate status protocol (OCSP) certificate checking for user authentication. If you enable this option, you muse enable OCSP in Unwired Server. This provider uses the values defined as part of the SSL security profile. Revoked certificates result in authentication failure when both of these conditions are met: • revocation checking is enabled • OCSP properties are configured correctly

Property	Description
Trusted certificate store	(Optional) The file containing the trusted CA certificates (import the issuer certificate into this certificate store). Use this property and `Store Password` property to keep the module out of the system trust store. The default Unwired Server system trust store is *`SUP_HOME`*`\Servers\UnwiredServer\Repository\Securitytruststore\truststore.jks`. If you do not specify a store location:: • Unwired Server checks to see if a store used by the JVM (that is, the one defined by the `javax.net.ssl.trustStoreType` system property. • If the system property is not defined, then this value is used: `${java.home}/lib/security/jssecacerts` • If that location also doesn't exist, then this value is used: `${java.home}/lib/security/cacerts` **Note:** This property is required only if Validate certificate path is set to true.
Trusted certificate store password	(Optional) The password required to access the trusted certificate store. For example, import the issuer of the certificate you are trying to authenticate into the shared JDK cacerts file and specify the password using this property. **Note:** This property is required only if Validate certificate path is set to true. However, you do not need to configure this value if the default is used. The default value is the value of the `javax.net.ssl.trustStorePassword` property.
Trusted certificate store provider	(Optional) The keystore provider. For example, "SunJCE." **Note:** This property is required only if Validate certificate path is set to true. However, you do not need to configure this value if the default is used. The default value is the value of the `javax.net.ssl.trustStoreProvider` property. If it is not defined, then the most preferred provider from the list of registered providers that supports the specified certificate store type is used.

Property	Description
Trusted certificate store type	(Optional) The type of certificate store. For example, "JKS." **Note:** This property is required only if Validate certificate path is set to true. However, you do not need to configure this value if the default is used. The default value is the value of the `javax.net.ssl.trustStore` property. If this value is not defined, then default value is the keystore type as specified in the Java security properties file, or the string "jks" (Java keystore) if no such property exists.
Validate certificate path	If true (the default), performs certificate chain validation of the certificate being authenticated, starting with the certificate being validated. Verifies that the issuer of that certificate is valid and is issued by a trusted certificate authority (CA), if not, it looks up the issuer of that certificate in turn and verifies it is valid and is issued by a trusted CA. In other words, it builds up the path to a CA that is in the trusted certificate store. If the trusted store does not contain any of the issuers in the certificate chain, then path validation fails. For information about adding a certificate to the truststore, see *Using Keytool to Generate Self-Signed Certificates and Keys* in *Security*.

See also

• *Adding a certificateValidationLoginModule for DCN Mutual Authentication* on page 142

HTTP Basic Authentication Properties

The HttpAuthenticationLoginModule provider authenticates the user with given credentials (user name and password) against the secured Web server using a GET against a URL that requires basic authentication, and can be configured to retrieve a cookie with the configured name and add it to the JAAS subject to facilitate single sign-on (SSO).

Configure this provider to authenticate the user by:

- Using only the specified user name and password.
- Using only the specified client value or values.
- Attempting token authentication. If that fails, revert to basic authentication using the supplied user name and password. You may find this helpful when using the same security configuration for authenticating users with a token, such as device users hitting the network edge, and when DCN requests from within a firewall present a user name and password but no token.

Note: The HttpAuthenticationLoginModule allows token validation by connecting to an HTTP server capable of validating the token specified in the HTTP header and cookie set in the session.

Table 10. HttpAuthenticationLoginModule Configuration Options

Configuration Option	Default Value	Description
URL	None	The HTTP or HTTPS URL that authenticates the user. For SSO, this is the server URL from which Unwired Server acquires the SSO cookie/token.
Disable certificate validation	False	(Optional) The default is false. If set to true, this property disables certificate validation when establishing an HTTPS connection to the SWS using the configured URL. Set to true only for configuration debugging.
SSO cookie name	None	(Optional) Name of the cookie set in the session between the LoginModule and the secured Web server, and holds the SSO token for single sign-on. The provider looks for this cookie in the connection to the secured Web server. If the cookie is found, it is added to the authenticated subject as a named credential. The authentication provider ignores the status code when an SSO cookie is found in the session; authentication succeeds regardless of the return status code.
Roles HTTP header	None	(Optional) The name of an HTTP header that the server may return. The header value contains a comma-separated list of roles to be granted.

Configuration Option	Default Value	Description
Successful connection status code	200	HTTP status code interpreted as successful when connection is established to the secured Web server.
HTTP connection timeout interval	60 seconds	The value, in seconds, after which an HTTP connection request to the Web-based authentication service times out. If the HTTP connection made in this module (for either user authentication or configuration validation) does not have a timeout set, and attempts to connect to a Web-based authentication service that is unresponsive, the connection also becomes unresponsive, which could potentially cause Unwired Server to become unresponsive. Setting the timeout interval ensures that authentication failure is reported without waiting indefinitely for the server to respond.
SendClientHttpValuesAs	None	Comma-separated list of strings that indicate how to send ClientHttpValuesToSend to the HTTP server. For example: `SendClientHttpValuesAs=header:`*`header_name`*`, cookie: `*`cookie_name`* This property does not apply if the user is to be authenticated using only the supplied user name and password .

Configuration Option	Default Value	Description
ClientHttpValuesToSend	None	A comma-separated list of client HTTP values to be sent to the HTTP server. For example: `ClientHttpValuesToSend=client_personalization_key, client_cookie_name` Set this property if you are using token authentication. Setting this property triggers token authentication. Only token authentication is attempted, unless TryBasicAuthIfTokenAuthFails is configured to true in conjunction with ClientHttpValuesToSend. This property does not apply if the user is to be authenticated using only the supplied user name and password .

Configuration Option	Default Value	Description
SendPasswordAsCookie	None	Deprecated. Use only for backward compatibility. New configurations should configure token authentication using SendClientHttpValuesAs and ClientHttpValuesToSend. Sends the password to the URL as a cookie with this name. If not specified, the password is not sent in a cookie. This property is normally used when there is a cookie-based SSO mechanism in use (for example, SiteMinder), and the client has put an SSO token into the password. The token can be propagated from the personalization keys and HTTP header and cookies to the secured Web server without impacting the password field.
TryBasicAuthIfTokenAuthFails	False	Specifies whether the provider should attempt basic authentication using the specified user name and password credentials if token authentication is configured and fails. This property is applicable only if token authentication is enabled. This property does not apply if the user is to be authenticated using only the supplied user name and password .

Configuration Option	Default Value	Description
UsernameHttpHeader	None	HTTP response header name returned by the HTTP server with the user name retrieved from the token. Upon successful authentication, the retrieved user name is added as a SecNamePrincipal. This property does not apply if the user is to be authenticated using only the supplied user name and password .
regexForUsernameMatch	None	Regular expression used for matching the supplied user name with the user name returned by the HTTP server in the UsernameHttpHeader. The string "{username}" in the regex is replaced with the specified user name before using it. If specified, it matches the user name retrieved from the UsernameHttpHeader to the user name specified in the callback handler. If the user names do not match, authentication fails. If the user names match, both the specified user name and the retrieved user name are added as SecNamePrincipals to the authenticated subject. This property does not apply if the user is to be authenticated using only the supplied user name and password .

Configuration Option	Default Value	Description
TokenExpirationTimeHttpHeader	None	HTTP response header name that is returned by the HTTP server with the validity period of the token in milliseconds since the start of January 1, 1970. If the header is returned in the HTTP response from the secured Web server, the token is cached for the duration it remains valid unless TokenExpirationInterval is also configured. If this response header is not returned with the token, it might result in unintended use of the token attached to the authenticated context even after it has expired. This property does not apply if the user is to be authenticated using only the supplied user name and password .

Configuration Option	Default Value	Description
TokenExpirationInterval	0	Specifies the interval, in milliseconds to be deducted from the actual expiration time returned in TokenExpirationTimeHttpHeader. This ensures that the token credential retrieved from the authenticated session remains valid until it is passed to the SWS for single sign-on to access MBOs. **Note:** This property does not apply if the user should be authenticated using only the supplied user name and password. **Note:** If the configured TokenExpirationInterval value exceeds the amount of time the token is valid, authentication by the HttpAuthenticationLoginModule fails even if the token is validated successfully by the secured Web server.
CredentialName	None	Name to set in the authentication credential that contains the token returned in SSOCookieName. If this property is not configured, the SSOCookieName is set as the name of the token credential.

See also

- *Certificate Security Provider* on page 61
- *SAP SSO Token Security Provider* on page 61
- *HTTP Authentication Security Provider* on page 62

SAP SSO Token Authentication Properties

The SAPSSOTokenLoginModule has been deprecated. Use the HttpAuthenticationLoginModule when SAP SSO2 token authentication is required.

See also

- *Certificate Security Provider* on page 61
- *SAP SSO Token Security Provider* on page 61
- *HTTP Authentication Security Provider* on page 62

Preconfigured User Authentication Properties

The PreConfiguredUserLoginModule authenticates the Unwired Platform Administrator user whose credentials are specified during installations.

This login module is recommended only to give the Platform administrator access to Sybase Control Center so it can be configured for production use. Administrators are expected to replace this login module immediately upon logging in for the first time.

The PreConfiguredUserLoginModule:

- Provides role based authorization by configuring the provider com.sybase.security.core.RoleCheckAuthorizer in conjunction with this authentication provider.
- Authenticates the user by comparing the specified username/password against the configured user. Upon successful authentication, the configured roles are added as Principals to the Subject.

Table 11. PreConfiguredUserLoginModule properties

Property	Description
User name	A valid user name. Do not use any of these restricted special characters: `, = : ' " * ? &`.
Password	The encoded password hash value.

Property	Description
Roles	Comma separated list of roles granted to the authenticated user for role-based authorization. Platform roles include "SUP Administrator", "SUP Domain Administrator", and "SUP Helpdesk". Roles are mandatory for "admin" security configuration. For example, if you define "SUP Administrator" to this property, the login id in the created login module has Platform administrator privileges. The "SUP Helpdesk" role has the fewest privileges. If multiple roles are defined for this property, a role with more privileges ("SUP Administrator" or "SUP Domain Administrator") is used for authorizing users. **Note:** If you use other values, ensure you map Unwired Platform roles to the one you define here.

See also

- *Adding a Production-Grade Provider* on page 33
- *Adding a PreconfiguredUserLoginModule for HTTP Basic Authentication* on page 145
- *Mapping Unwired Platform Logical Roles to Physical Roles* on page 34

Audit Provider Properties

The security configuration for Sybase Unwired Platform includes an audit provider with three components: audit filter, audit formatter, and audit destination.

An auditor consists of one destination, one filter, and one formatter.

- The supported value for destination is `com.sybase.security.core.FileAuditDestination`. Optionally, you can develop a custom provider and configure it as the audit destination, formatter, and filter. See *CSI Audit Generation and Configuration*.
- The only supported value for the filter is `com.sybase.security.core.DefaultAuditFilter`.
- The only supported value for the formatter is `com.sybase.security.core.XmlAuditFormatter`.

For information on developing a custom provider and configuring it as the audit destination, formatter, and filter, see *Security API* in *Developer Guide: Unwired Server Runtime*.

For detailed information on the audit packages, see *Security Configuration* in *Developer Guide: Unwired Server Runtime > Management API*.

DefaultAuditFilter Properties

The audit filter component configures the resource classes for which the audit records should be routed to the associated destination.

Filter resource classes require a specific syntax. The audit token identifies the source for core audit requests of operations, such as auditing the results for authorization and authentication decisions, in addition to placing information such as active provider information into the audit trail. The audit records have their resource class prefixed by the prefix core. The CSI core is able to audit multiple items.

DefaultAuditFilter Configuration Properties

The property name default value description is:

```
(1)caseSensitiveFiltering false set to true to use case sensitivity
when matching resource classes and actions
(2)filter
default
value="(ResourceClass=core.subject,Action=authorization.role)
(ResourceClass=core.subject,Action=authorization.resource)
(ResourceClass=core.subject,Action=authentication)
(ResourceClass=core.subject,Action=logout)
(ResourceClass=core.profile)
(ResourceClass=providers.*)(ResourceClass=clients.*)"
description = the filter string that determines whether an audit
record should be written out to the audit destination.
```

Syntax

Filter resource classes consist of one or more filter expressions that are delimited by parenthesis (()). Square brackets ([]) denote optional values. The syntax is:

```
[key1=value [,key2=value...]].
```

The allowed keys are: `ResourceClass`, `Action`, or `Decision`.

This table describes core auditable items:

Resource Class	Action	Description	Attributes
provider	activate	Called when a provider is activated by CSI. The resource ID is the provider class name.	Generated unique provider identifier.

Resource Class	Action	Description	Attributes
subject	authentication.provider	The result of a provider's specific authentication request. Depending on the other providers active, the actual CSI request for authentication may not reflect this same decision. This resource class is not a provider-generated audit record. CSI core generates this audit record automatically after receiving the provider's decision. The resource ID is not used.	• Provider identifier • Decision (yes or no) • Failure reason (if any) • Context ID
subject	authentication	The aggregate decision after considering each of the appropriate provider's authentication decisions. This record shares the same request identifier as the corresponding authentication provider records. The resource ID is the subject identifier if authentication is successful.	• Decision (yes or no) • Context ID
subject	authorization.role.provider	The result of a provider's specific role authorization request. The resource ID is the subject ID.	• Provider identifier • Decision (yes, no or abstain) • Role name • Supplied subject identifier, if different from context subject • Context ID

Resource Class	Action	Description	Attributes
subject	authorization.role	The result of a resource-based authorization request. The resource ID is the subject ID.	• Resource name • Access requested • Decision (yes or no) • Supplied subject identifier, if different from context subject • Context ID
subject	logout	Generated when an authenticated context is destroyed. The resource ID is the subject ID.	• Context ID
subject	create.provider	Provider-level record issued for anonymous self-registration requests. The resource ID is the subject identifier.	• Provider identifier • Decision • Subject attributes
subject	create	Aggregate, generated when an anonymous self-registration request is made. The resource ID is the subject identifier.	• Decision • Subject attributes
subject	authorization.resource	The aggregate authorization decision, which is made after considering each of the appropriate provider's result. The resource ID is the subject ID.	• Resource ID • Access requested • Decision (yes or no) • Subject ID supplied, if different from context subject • Context ID

Examples

- **Example 1 –** enables auditing of all the CSI core resource classes that involve a deny decision:

```
(ResourceClass=core.*,Decision=Deny)
```

- **Example 2 –** enables auditing for all core resource classes where the action is the subject modification action:

```
Resource=core.*,Action=subject.modify.*)
```

FileAuditDestination Properties

The FileAuditDestination is a simple file-based provider that logs the audit records to a file which is rolled over upon reaching a specified size.

This provider can safely share access to a file between multiple instances as long as they are all in the same VM. To integrate with a customer's existing audit infrastructure, a custom audit destination provider can be developed and deployed. See *com.sybase.security.core.FileAuditDestination* in *Developer Guide: Unwired Server Runtime*.

Table 12. File Audit Destination Configuration Options

Configuration Option	Description
auditFile	The absolute path of the file to write the audit records.
encoding	The character encoding used when writing the audit data (default=UTF-8).
logSize	This option may be supplied to specify the maximum audit log file size before a rollover occurs.
compressionThreshold	This option may be supplied to specify the number of uncompressed audit log rollover files that are created, before compression is used to archive the audit data.
deleteThreshold	This option may be supplied to specify the number of audit log files that will be preserved. This value includes the main audit log, so a value of "3" allows an audit.log, audit.log.0 and audit.log.1 before deleting old logs.

Configuration Option	Description
errorThreshold	This option may be supplied to specify the maximum number of audit log files that are allowed. When this threshold is reached, an error occurs and all auditing fails. For example, with this value set to "3", audit.log, audit.log.0 and audit.log.1 will be created according to the maximum log size value. If another audit log rollover is triggered, then all audit operations will fail until one of the rollover files is removed. This value is mutually exclusive with the deletion threshold, and the smallest value of the two takes effect.

XMLAuditFormatter Properties

The audit formatter component formats an audit record from its component parts. An audit formatter is supplied to the active audit destination upon initialization, where the audit destination can use the formatter if required.

The default provider com.sybase.security.core.XmlAuditFormatter formats audit data into an XML record. The audit records generated by this provider are of the format

```
<AuditRecord Action="[action]" Decision="[decision]"
When="[timestamp]"> <Resource Class="[resource classname]"
ID="[resource id]" /> <Attribute Name="[attribute1 name]"
Value="[attribute1 value]" /> <Attribute Name="[Map attribute name]"
Key="[Map Key1 name]" Value="[Map value associated with the key1]" />
<Attribute Name="[Map attribute name]" Key="[Map Key2 name]"
Value="[Map value associated with the key2]" /> <Attribute
Name="[List attribute name]" Value="[List value1]" /> <Attribute
Name="[List attribute name]" Value="[List value2]" /> </AuditRecord>
```

Certificate and Key Management Utilities

Use the certificate management utilities to encrypt Unwired Server ports.

Launch these utilities from the command line; the certificate and key management utilities are not available from any other administration tool.

Use **createcert** and **createkey** for MobiLink and Ultralite server/client purposes (specific for replication payload protocol packages). For all other purposes, use **keytool** or the PKI system deployed to your environment.

Certificate Creation (createcert) Utility

Generates X.509 certificates or signs pregenerated certificate requests. This utility is located in *SUP_HOME*\Servers\SQLAnywhere*XX*\BIN*XX*.

You may choose to purchase certificates from a third party. These certificate authorities (CAs) provide their own tools for creating certificates. You can use **createcert** to create certificates for development and testing; you can also use it for production certificates.

Syntax

```
createcert [options]
```

Parameters

- **[*options*] –** these options are available through the **createcert** utility:

Option	Description
`-r`	Creates a PKCS #10 certificate request. **createcert** does not prompt for a signer or any other information used to sign a certificate.
`-s <filename>`	Signs the PKCS #10 certificate request that is in the specified file. The request can be DER or PEM encoded. **createcert** does not prompt for key generation or subject information.

Note: To create a signed certificate, use **createcert** without options. To break the process into two separate steps, for example so one person creates a request and another person signs it, the first person can run **createcert** with `-r` to create a request and the second person can sign the request by running **createcert** with `-s`.

When you run **createcert**, you are asked for all or some of this information, depending on whether you specified `-r`, `-s`, or neither.

- `Choose encryption type` – select RSA.
- `Enter RSA key length (512-16384)` – this prompt appears only if you chose RSA encryption. Specify a length between 512 bits and 16384 bits.
- Subject information – enter this information, which identifies the entity:
  - `Country Code`
  - `State/Province`
  - `Locality`
  - `Organization`
  - `Organizational Unit`
  - `Common Name`
- (Optional) `Enter file path of signer's certificate` – supply a location and file name for the signer's certificate. If you supply this information, the

generated certificate is a signed certificate. If you do not supply this information, the generated certificate is a self-signed root certificate.

- `Enter file path of signer's private key` – supply a location and file name to save the private key associated with the certificate request. This prompt appears only if you supplied a file in the previous prompt.
- `Enter password for signer's private key` – supply the password that was used to encrypt the signer's private key. Supply this password only if the private key was encrypted.
- (Optional) `Serial number` – supply a serial number. The serial number must be a hexadecimal string of 40 digits or less. This number must be unique among all certificates signed by the current signer. If you do not supply a serial number, **createcert** generates a GUID as the serial number.
- `Certificate will be valid for how many years (1-100)` – specify the number of years for which the certificate is valid. After this period, the certificate expires, along with all certificates it signs.
- `Certificate Authority (y)es or (n)o` – indicate whether this certificate can be used to sign other certificates. The default value is no.
- `Key usage` – supply a comma-separated list of numbers that indicate the ways in which the certificate's private key can be used. The default, which depends on whether the certificate is a certificate authority, should be acceptable for most situations.
- `File path to save request` – this prompt appears only if you specify the -r option. Supply a location and file name for the PCKS #10 certificate request. Supply a location and file name in which to save the certificate. The certificate is not saved unless you specify a location and file name.
- `Enter file path to save private key` – supply a location and file name in which to save the private key. Enter a password to protect private key. Optionally, supply a password with which to encrypt the private key. If you do not supply a password, the private key is not encrypted. This prompt appears only if you supplied a file in the previous prompt.
- `Enter file path to save identity` – supply a location and file name in which to save the identity. The identity file is a concatenation of the certificate, signer, and private key. This is the file that you supply to the server at start-up. If the private key was not saved, **createcert** prompts for a password to save the private key. Otherwise, it uses the password provided earlier. The identity is not saved unless you provide a file name. If you do not save the identity file, you can manually concatenate the certificate, signer, and private key files into an identity file.

Examples

- **Example 1: –** creates a self-signed certificate. No file name is provided for the signer's certificate, which makes it a self-signed root certificate.

```
SUP_HOME\UnwiredPlatform\Servers\SQLAnywhereXX\BINXX>createcert
SQL Anywhere X.509 Certificate Generator Version xx.xx.xx.xx
```

```
Enter RSA key length (512-16384): 1024
Generating key pair...
Country Code: US
State/Province: CA
Locality: Dublin
Organization: MyCompanyCA
Organizational Unit: PTO
Common Name: MyCompanyCA
Enter file path of signer's certificate:
Certificate will be a self-signed root
Serial number [generate GUID]:<enter>
Generated serial number: 3f52ee68c8604e48b8359e0c0128da5a
Certificate valid for how many years (1-100): 10
Certificate Authority (Y/N) [N]: Y
1.  Digital Signature
2.  Nonrepudiation
3.  Key Encipherment
4.  Data Encipherment
5.  Key Agreement
6.  Certificate Signing
7.  CRL Signing
8.  Encipher Only
9.  Decipher Only
Key Usage [6,7]: <enter>
Enter file path to save certificate: rsa_root.crt
Enter file path to save private key: rsa_key.key
Enter password to protect private key: <MyPwd>
Enter file path to save identity: id.pem
```

- **Example 2: Generating an enterprise root certificate –** to generate an enterprise root certificate (a certificate that signs other certificates), create a self-signed root certificate with a CA. The procedure is similar to Example 1. However, the response to the CA prompt should be `yes` and choice for roles should be option `6,7` (the default).

```
Certificate Authority (Y/N) [N]: y
1.  Digital Signature
2.  Nonrepudiation
3.  Key Encipherment
4.  Data Encipherment
5.  Key Agreement
6.  Certificate Signing
7.  CRL Signing
8.  Encipher Only
9.  Decipher Only
Key Usage [6,7]: 6,7
```

Key Creation (createkey) Utility

Creates an RSA key pairs for use with Unwired Server end-to-end encryption. This utility is located in *SUP_HOME*\Servers\SQLAnywhere*XX*\BIN*XX*.

Syntax

```
createkey
```

When you run **createkey**, you are prompted for this information:

- `Choose encryption type` – choose RSA.
- `Enter RSA key length (512-16384)` – this prompt appears only if you chose RSA encryption. Choose a length between 512 bits and 16384 bits.
- `Enter file path to save public key` – specify a file name and location for the generated PEM-encoded public key. This file is specified on the MobiLink client by the e2ee_public_key protocol option.
- `Enter file path to save private key` – specify a file name and location for the generated PEM-encoded private key. This file is specified on the MobiLink server via the e2ee_private_key protocol option.
- `Enter password to protect private key` – optionally, supply a password with which to encrypt the private key. The private key is not encrypted if you do not supply a password. This password is specified on the MobiLink server via the e2ee_private_key_password protocol option.

Examples

- **Example –** creates an RSA key pair:

```
>createkey
SQL Anywhere Key Pair Generator Version 11.0.0.2376
Enter RSA key length (512-16384): 2048
Generating key pair...
Enter file path to save public key: rsa_key_public.key
Enter file path to save private key: rsa_key_private.key
Enter password to protect private key: pwd
```

Truststore and Keystore Properties

The *SUP_HOME*\SCC-*XX*\services\Messaging\lib\eas\lib\Repository\Server\EmbeddedJMS\Instance\com\sybase\djc\server\ApplicationServer\EmbeddedJMS.properties file contains properties for the truststore and keystore that you can configure. While Sybase Control Center uses the same keystore and truststore location as Unwired Server, This file only configures the keystore/truststore for the Sybase Control Center Windows service.

Change the default properties for:

Property	Default	Description
keyStore	*SUP_HOME*/Servers/UnwiredServer/Repository/Security/keystore.jks	The default location of the keystore used by Sybase Control Center.

Property	Default	Description
keyStorePassword	changeIt	The password used to unlock the keystore.
trustStore	*SUP_HOME*/Servers/UnwiredServer/Repository/Security/truststore.jks	The default location of the truststore used by Sybase Control Center. The truststore is used when Sybase Control Center makes an out-bound connection over SSL to another server with a server certificate. Sybase Control Center checks that the server certificate is in the truststore, or is signed by a CA certificate in the truststore.
trustStorePassword	changeIt	The password used to unlock the truststore.

Port Number Reference

Change Sybase Unwired Platform component port numbers after installation, if necessary.

Proceed with caution when changing port numbers because the change might impact other configuration files that point to that port. You need to be aware of the default Sybase Control Center port numbers so you do not accidentally use these ports when you change Unwired Platform ports. You can change some Sybase Control Center default ports, but, in some cases, you should not.

Note: To make Unwired Server port number changes, temporarily stop the other service consuming those ports. Use Sybase Control Center to make the changes, then restart Unwired Server.

Note: Port numbers 5701, 5702, and 5011 should be reserved ports.

Port	Description	Default Port	Instructions for Changing
Data tier (CDB) server	Port number for the data tier that manages transactions between the enterprise information system and mobile devices.	5200	Do not change the CDB port.

Port	Description	Default Port	Instructions for Changing
Management ports	IIOP port number on which the Unwired Server listens for Sybase Control Center administration requests.	2000 2001 for secure management (default)	Default is recommended. No change is required.
HTTP ports	HTTP port number on which Unwired Server listens for: • Data change notification (DCN) events (server authentication). • HTTP channel notification (mutual authentication).	8000 for HTTP 8001 for HTTPS	Configure in Sybase Control Center by selecting **Configuration**, clicking the **Web Container** tab and entering a new DCN port or secure DCN port, as required. See *Configuring Security Profiles* in *Sybase Control Center for Sybase Unwired Platform* online help.
Synchronization	Port numbers on which Unwired Server synchronizes data between the enterprise information system and mobile devices. Messaging port uses a proprietary encryption method, so communication is always encrypted.	2480 for replication 5001 for messaging	Configure in Sybase Control Center by selecting **Configuration**. In the **Components** tab, select **Replication** or **Messaging**, click **Properties** and enter a new synchronization port, as required. **Note:** If there is a conflict for port 2480 or 2481, Unwired Server will not start, and you cannot use Sybase Control Center to modify them. To correct the problem, you must temporarily stop the service that uses the conflicting port, then start Unwired Server. For replication payloads, see *Configuring Replication Subscription Properties* in *Sybase Control Center for Sybase Unwired Platform* online help. For messaging payloads, see *Configuring Messaging Properties* in *Sybase Control Center for Sybase Unwired Platform* online help.

Port	Description	Default Port	Instructions for Changing
Messaging server administration	Port number for the messaging service for Sybase messaging clients.	5001 for administration services	Cannot be changed in Sybase Control Center. Use the *SUP_HOME*\Servers\Messaging Server\Bin\AdminWebServicesTool.exe command line tool to change the messaging service Web service port. This tool has built in online help describing how to use the tool. From the command prompt run: *SUP_HOME*\Servers\Messaging Server\Bin > AdminWebServicesTool.exe set=<*port*> restart

Port	Description	Default Port	Instructions for Changing
Sybase Control Center	Additional default port numbers of which to be aware, when modifying port numbers.	9999 for default RMI agent port 2100 for default JMS messaging service port 3638 for default Sybase Control Center repository database port 8282, 8283 for default Web container ports	• 9999 – default RMI agent port. The port is set in: *SCC_HOME*\services\RMI\service-config.xml • 2100 – default JMS messaging service port. The port is set in: *SCC_HOME*\services\Messaging\service-config.xml • 3638 – default Sybase Control Center repository database port. The default port is set in: *SCC_HOME*\services\SccSADataserver\service-config.xml • 8282, 8283 – default Web container ports. The default ports are set in: *SCC_HOME*\services\EmbeddedWebContainer\service-config.xml Before you make any changes to these files, stop Sybase Control Center X.X service. Start the service after you complete the changes. If any of the subsystems fail to start, check the Sybase Control Center agent.log for error messages.
Relay server	Port numbers on which Relay Server listens for requests.	80 for the HTTP port 443 for the HTTPS port	Change the value for the cluster in Sybase Control Center for Unwired Platform. You can then generate the file and transfer it to the corresponding Unwired Server host. See *Setting Relay Server General Properties* in *Sybase Control Center for Sybase Unwired Platform*.

Port	Description	Default Port	Instructions for Changing
Unwired Platform reserved	Port numbers reserved for internal use by Unwired Platform components	2638 4343 6001 5500 8002 27000	Do not use these special port numbers for any purpose. These ports differ from Windows reserved ports (1-1023). **Note:** Even if the installer does not detect a conflict at install time, Windows may later use ports in the 1024-64K range for other purposes. Read Microsoft documentation to determine how to reserve Unwired Platform ports. Otherwise, you may experience intermittent problems when starting up platform services due to Windows using these ports for some other purpose at the same time.

Index

A

Accessing LDAP roles 59
Active Directory nested groups 59
administrators
    domain administrator role 35
    help desk role 35
    platform administrator role 34
alias, certificate 88
applications 127
audit destination 182
audit filter 179
audit formatter 183
authentication
    configuring for Unwired Server 33
    how it works 52
authentication cache timeouts 54
AuthenticationScope 57
authorization
    DCNs 137
    Push notifications 137

C

cache timeouts 54
certificate alias 88
certificate creation CLU 184
CertificateAuthenticationLoginModule authentication module
    for SAP single sign-on and X.509 61, 94, 164
checkImpersonation 74
ClientHttpValuesAsNamePrincipals 72
ClientHttpValuesAsRolePrincipals 72
ClientValuePropagatingLoginModule 72
command line utilities
    createkey 186
communication ports
    SSL encryption 108
connection templates, creating 83
connections, creating 83
control flag 99
controlFlag 99
createcert command line utility 184
createkey utility 186
creating a SAP JCo connection
    for SAP single sign-on 84
CSI security
    troubleshooting 101

D

data
    encrypting administration data 17, 45
DCN 147
documentation roadmap 1
domain administrator 35
domains
    creating 49
    enabling 49

E

EIS
    Push operations 137
encrypting
    administration data 17, 45
encryption certificates
    Unwired Server administration 41, 103

G

generating X.509 certificates
    for SAP single sign-on 85

H

help desk 35
HTTP cookies 72
HTTP headers 72
HttpAuthenticationLoginModule authentication module
    for SAP single sign-on 93

I

intrusion detection/prevention software 28

K

key creation utility 186

keytool.exe 60

L

LDAP
    configuration properties 153
    configuring Unwired Server to use 33
    role computation 57
    stacking providers 58
LDAP nested groups 59
LDAP security provider
    modules available 56
LDAP SSL configuration 60
LDAP trees
    multiple 57
logical roles
    DCNs 137
    Push notifications 137

M

Manual registration 127
mapping roles
    dynamically 65
monitoring 149
    user security 150
monitoring data
    reviewing 149
multi tenancy
    tenancy strategy 48
multiple LDAP trees 57

N

NamedCredential 72
network edge 71

P

performance properties, configuring for server 50
platform administrators 34
platform security introduction 1
port numbers 188
preparing the SAP server
    SAP single sign-on 92
properties
    security provider configuration 153
providers
    authentication, how it works 52
    underlying technologies 55

R

reauthentication avoidance 54
roles
    computing 57
    mapping 65

S

SAP single sign-on
    creating a SAP JCo connection 84
    deploying packages and bundles 80
    generating X.509 certificates 85
    HttpAuthenticationLoginModule
        authentication module 93
    in an Unwired Server cluster 96
    preparing the SAP server 92
    SAPSSOTokenLoginModule authentication
        module 61
    SAPSSOTokenLoginModule authentication
        properties 176
    stacking login modules 100
SAP single sign-on with X.509
    CertificateAuthenticationLoginModule
        authentication module 61, 94, 164
SAP/R3 properties 86
SAPSSOTokenLoginModule authentication
    module
    for SAP single sign-on 61
    properties 176
security
    monitoring 150
security by tiers 2
security configuration
    creating 53
security configurations
    for administration 33
    overview 8
    troubleshooting 101
security profile
    SSL certificates 107
security profiles 108
    communication port 108
    management port 108
security provider configuration properties 153
security providers
    troubleshooting 101

server configuration
    system performance properties 50
single sign-on 71
single sign-on task flow 147
SiteMinder authentication 65
SiteMinder authentication cache timeout 67
SiteMinder security configuration 68
SiteMinder single sign-on 66
SiteMinder SSO 66
SiteMinder Web agent 68
Skipping LDAP role lookups 60
SkipRoleLookup 60
SOAP Web Services properties 88
SSL
    mutual authentication 88
SSL certificates 107
SSL encryption
    security profile 108
SSL keystore 107
SSL truststore 107
SSO 71
SSO integration 71
stacking LDAP modules 58
stacking login modules
    for SAP single sign-on 100
SunOne nested groups 59
SUP DCN User 137
SUP Push User 137
system data, reviewing 149

T

token expiry 54

U

Unwired Platform administrators 34
Unwired Server administration
    replacing default encryption certificates 41, 103
Unwired Server cluster
    SAP single sign-on 96
users
    security statistics 150